General Services Administration
Federal Acquisition Service
Integrated Award Environment

INTEGRATED AWARD ENVIRONMENT (IAE) AGILE FRAMEWORK

August 24, 2015

Version 1.1

U.S. General Services Administration

Document Control

Change Record

This section provides control for the development and distribution of revisions to the document.

Version	Date of Issue	Authors	Brief Description of Change
1.0	Feb 2015	Government and Technical Governance Agile Team Members	Initial draft
1.1	August 2015	Government and Technical Governance Agile Team Members	Removed Hardening Sprint, Hotfix references Reorganized content for readability Aligned content to IAE processes Updated agile framework process flow graphics Added estimation techniques Added Definition of Ready section

Contents

Figures

Tables

List of Attachments

IAE Agile Metrics	V1.5	7/2/2015
IAE CSP Architecture	v1.3	7/20/2015

1 Executive Summary

1.1 Scope

The Integrated Award Environment (IAE) Agile Framework document describes the Core Values, Agile and Architecture integration, Agile Implementation Framework and Governance mechanisms for enterprise Agile adoption for the Integrated Award Environment.

The purpose of the IAE Agile Framework is to share principles, standards, processes, practices and guidance on the Agile approach for the IAE program. This document will cover the governance processes for the portfolio, program and teams, key roles and responsibilities, metrics, and measurements for consistent adoption of the framework for the enterprise.

1.2 IAE Agile Framework Vision

The IAE Agile Framework Vision is to design and deploy the new IAE by executing business strategies with a collaborative team spirit, ongoing software delivery, continual stakeholder engagement, and a relentless desire to improve upon the status quo.

The IAE Agile Framework is intended to:

- Implement adoption of Agile practices across the enterprise to achieve alignment throughout the organization–stakeholders and vendors alike–to support the overall mission of the organization.
- Create common definitions that are used consistently at the enterprise level to minimize confusion over usage of terms in the IAE context.
- Establish a lean and effective IAE governance process that creates the foundation for consistent and reliable reporting.
- Provide a blueprint for Agile that creates transparency to enable communication and coordination within and across the enterprise.
- Ensure a process that scales investment manageability, lowers risk of project failure, shortens the time to realize value, and allows agencies to better adapt to changing needs.

At the core, the IAE Agile Framework is designed to adopt Agile principles and enable execution at the enterprise scale to support IAE's mission. It is our intent to continue to inspect and adapt the implementation of the IAE Agile Framework to make it work for the Program needs.

The IAE Governance Model (Figure 1) contains the high-level governance model for the Agile process. The IAE Governing Body represents the key stakeholders across the Federal Government (the Award Committee for E-Government and its related subcommittees). The other teams are described in detail throughout this document.

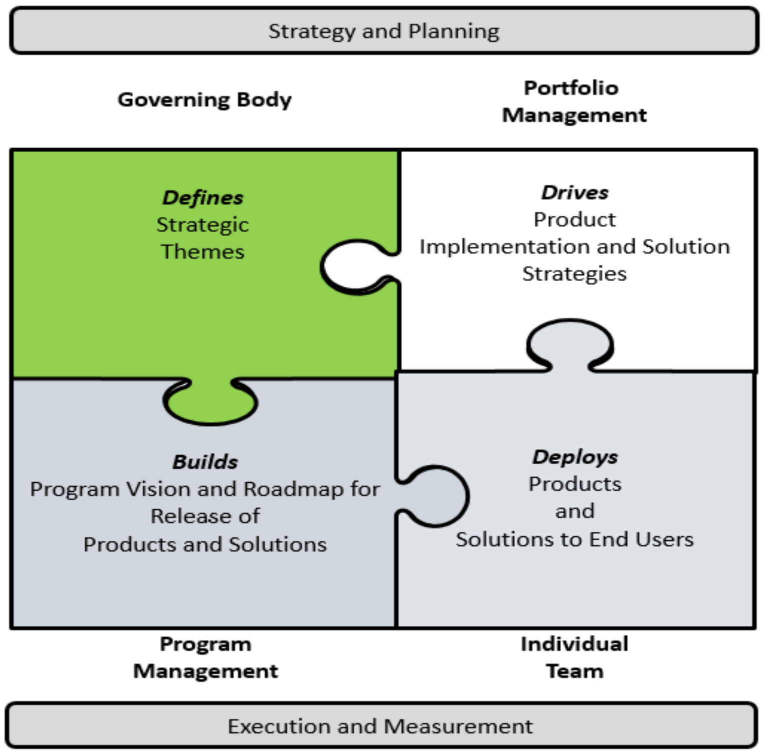

Strategy and Planning
Governing Body
Portfolio Management
Defines Strategic Themes
Drives Product Implementation and Solution Strategies
Builds Program Vision and Roadmap for Release of Products and Solutions
Deploys Products and Solutions to End Users
Program Management
Individual Team
Execution and Measurement

FIGURE 1: IAE GOVERNANCE MODEL

1.3 IAE AGILE FRAMEWORK STRATEGY REFERENCES

The IAE Agile Framework documentation/implementation strategy is based on the content listed below, and additional content will be incorporated in upcoming revisions of this document.

- U.S. Digital Services Playbook
- The TechFAR Handbook for Procuring Digital Services Using Agile Processes
- Sketching with Code: Protosketching
- Open Source for Good Government
- Scaled Agile Framework (SAFe)

2 IAE Agile Principles and Practices

In 2001, a group of thought leaders from the software industry established a common set of overarching values and principles. They are succinctly summarized in the Agile Manifesto, which goes as follows:

We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile development is the practice of designing and releasing software Features in frequent intervals. Rather than releasing a final version of a software product at one time, Agile teams are continuously releasing new iterations of a product, adding new Features and refining the user experience based on constant customer feedback. Using Agile practices, organizations can enjoy a range of advantages beyond the traditional methods of software development.

- **Transparency** – Clearer / better expectation of what is happening and when it is happening
- **Collaboration** – Open invitation to attend team planning and stand-ups and engaging teams
- **Prioritization** – Involved in priority setting – which stories get worked, in what order
- **Acceptance** – Help groom (further define) the story – what is it you really want done
- **Business Value** – Incremental business value delivery

2.1 IAE Collaboration Principles

The IAE Agile Framework is supported by the following collaboration principles:

- We are transparent
- We are curious
- We are team-oriented
- We are respectful
- We work flatter across boundaries
- We have honest conversations
- We are agile, risk-attentive, and knowledge-hungry
- We listen to understand
- We value face-to-face conversation
- We are empowered and forthcoming

As Agile adoption has evolved for solutions delivery, the processes and practices have matured to support Agile development at an enterprise scale.

2.2 Agile Engineering Practices

IAE implementation must follow consistent standards and practices to support development and continuous delivery of high quality products. Code quality is achieved by using Agile Engineering Practices that are inspired from the Extreme Programming (XP) and Lean methodologies. Benefits of high quality code include:

- Higher customer satisfaction
- Stable code that can respond to business changes
- Development scalability
- Higher development velocity
- Ability to innovate

For a more detailed description of Agile engineering practices, see Appendix A8.

2.3 IAE Architecture Practices and Principles

IAE adheres to SAFe architecture practices that promote "every team deserves to see the bigger picture" and "every team is empowered to design their part."

IAE Agile Framework and Architecture are integrated very closely. This framework adheres to the following architecture principles:

- Be open (source code, data, Application Program Interfaces (APIs))
- Treat data as an asset
- Use continuous Improvement to drive Innovation
- Provide an effective user experience for all stakeholders
- Ensure that business transactions are time- and cost-measurable
- Treat security as foundational
- Build value over maintaining status quo

The SAFe Agile Architecture Principles elaborate on these concepts and are provided in Appendix A10.

3 IAE Agile Framework Key Concepts

The IAE Agile Framework uses Agile best practices derived from multiple Agile methodologies, including Scrum, Kanban, Lean, XP and a widely acknowledged Scaled Agile Framework (SAFe)® recognized for applying Lean-Agile practice at the enterprise scale. The IAE Agile Framework recognizes that there are several Agile methods that can be used successfully for delivery of software projects. The IAE Agile Framework recommends using Scrum and Agile Engineering Practices for Agile Teams. Scrum is an iterative and incremental Agile software development framework for managing product development. It defines a flexible, holistic product development strategy where a development team works as a unit to reach a common goal. Agile Engineering Practices support the development of high quality software.

3.1 Value Streams and the Agile Release Train

Value stream is a metaphor that describes the cadence-based process of building capabilities that provide a rhythmic flow of value to an organization, business and end user. Value streams are realized by the Agile Release Train (ART or train), which is a virtual entity, a team of Agile Teams led by a Release Train Engineer (RTE).

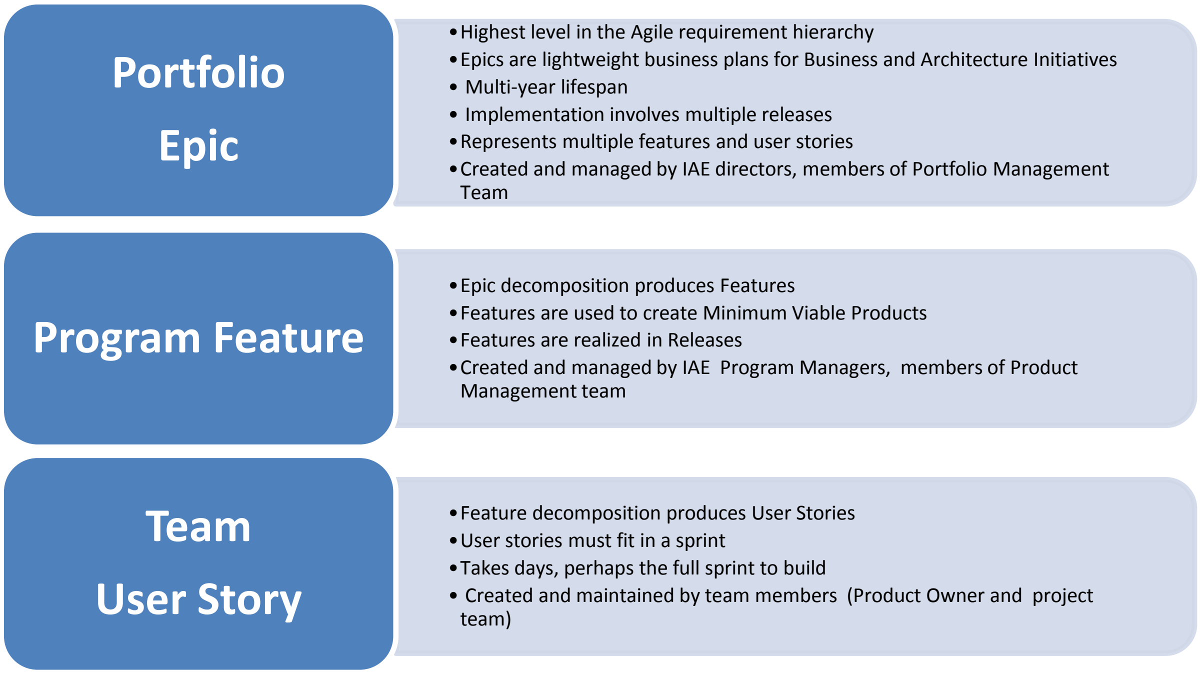
The train serves as the program-level value delivery mechanism. Each train has one Value Stream assigned to it. Each train also has the dedicated resources necessary to continuously define, build and test system-level solutions in a specific cadence; for IAE, the cadence comprises two-week sprints and four releases per year.

ART Principles:

- Release Increments are available at regular intervals for customer preview, internal review and system level Quality Assurance (QA).
- Frequent periodic planning and release dates for the solution are fixed (dates are fixed; quality is fixed; scope is variable).
- Teams develop to a common iteration period.
- Short-term and long-term objective milestones are established.
- Continuous system integration is implemented at the top, including system level as well as at the feature and component levels.
- Architecture Runway includes certain infrastructure components (e.g., common interfaces, infrastructure, system development kits, common installs, user stores and licensing utilities) must track ahead of the features that depend on them.

3.2 IAE Agile Requirements Model

The IAE Agile Framework promotes building solutions in an iterative and incremental manner. For this purpose, the framework will support a requirements model consisting of Epic → Feature → User Story creation to ensure that solutions are built to meet business needs effectively. Figure 2, Epic Decomposition (High Level), represents the relationship.


Portfolio Epic

- Highest level in the Agile requirement hierarchy
- Epics are lightweight business plans for Business and Architecture Initiatives
- Multi-year lifespan
- Implementation involves multiple releases
- Represents multiple features and user stories
- Created and managed by IAE directors, members of Portfolio Management Team

Program Feature

- Epic decomposition produces Features
- Features are used to create Minimum Viable Products
- Features are realized in Releases
- Created and managed by IAE Program Managers, members of Product Management team

Team User Story

- Feature decomposition produces User Stories
- User stories must fit in a sprint
- Takes days, perhaps the full sprint to build
- Created and maintained by team members (Product Owner and project team)

FIGURE 2: EPIC DECOMPOSITION (HIGH LEVEL)

3.2.1 EPICS

Epics are enterprise initiatives that are large enough such that their development could span multiple releases. There are business Epics (customer-facing) and architectural Epics (technology solutions).

At the Portfolio level, ideas are elaborated as Epic Value Statements (***Refer to Appendix A2, Epic Value Statement)***. Portfolio Epics can go across multiple ARTs. Proposed Epics are reviewed and analyzed by the Portfolio Management Team.

The Governing body approves initiatives, and the Portfolio Management team prepares Portfolio Epics using value statements, providing success criteria for each Epic, which resides in the Portfolio Backlog.

Portfolio Epics that are approved and assigned to an ART are called Program Epics. Epics approved for implementation are owned by an Epic Owner.

3.2.2 FEATURES

Epics decompose into Features, the next smallest requirement artifact. They are maintained in the Program Backlog and are sized to fit within one release. They can originate at the Program level, or they can derive from Epics defined at the Portfolio and Program levels. IAE has added a level below Feature, called Sub-Feature, which is simply a way of breaking down Features into smaller pieces of work. Features may or may not be broken down into Sub-Features.

A Features and Benefits Matrix (FAB) will be used to describe each Feature. Each Feature shall contain:

- Feature Name
- Feature Benefit: For the user and the organization

- Acceptance Criteria: To describe when the Feature is complete

Features are typically described using an action verb followed by a short phrase, for example “Create Payments via PayPal,” and a brief description of the benefit. Features are prioritized using the principle of Weighted Shortest Job First (WSJF). The Change Control Board approves Features to be implemented in each upcoming release.

3.2.3 User Stories

Features/Sub-Features decompose into User Stories, which should be constructed in such a way that they can be completed within one sprint. User Stories typically follow the pattern “As a <*persona*>, I can <*activity*> so that <*business value*>*”* and are primarily used to describe intent. User Stories do not contain implementation details. User Stories must contain acceptance criteria. Acceptance criteria are set of requirements that must be completed for the story to be finished and let the team know that the story is ‘Done’. In addition to being derived from Features, stories can also originate at the team level. User Stories can be created by anyone at the team level. During Sprint Planning, the team moves User Stories from the Team Backlog into the Sprint Backlog for implementation.

3.3 Backlog Management

In Agile, the term “backlog” refers to the requirements repository, and managing the backlog is a crucial element of the Agile Software Development Life Cycle (SDLC). The backlog contains three levels of requirements: Epics, Features and Stories which, along with the backlog itself, align with the three levels of the enterprise: Portfolio, Program and Team.

Portfolio-level Epics are prioritized, and as development team capacity becomes available, the highest priority Epics are decomposed into Program Epics and then Features at the Program level. Releases are planned quarterly, and features are estimated, prioritized and assigned to releases. Once a feature is assigned to a release, it is decomposed into user stories and tasks and then implemented in a series of two-week sprints.

The IAE Agile Framework has the following Backlogs that hold the requirement artifacts, as shown in Table 1:

TABLE 1: BACKLOG MANAGEMENT

Level	Backlog	Artifact	Scope	Duration	Responsible
Portfolio	Portfolio Backlog	Epics with Value Statements	Portfolio Epic can span multiple ARTs	An Epic spans multiple Releases	Portfolio Management Team
Program	Program Backlog	Program Epics with Value Statements, decomposed to Features	Program Epics and features are allocated to an ART	A Feature is realized in a Release	Product Management Team
Team	Team Backlog	User Stories with Acceptance Criteria	Team backlog	A User Story is implemented in a Sprint	Product Owner(s)
	Sprint Backlog	User Stories with Acceptance Criteria	Sprint backlog	A User Story is implemented in a Sprint	Product Owner(s)

3.3.1 PORTFOLIO BACKLOG

The Portfolio Backlog is aligned with the Portfolio level of the enterprise and is the single definitive repository for all upcoming work anticipated to achieve the strategic vision for the enterprise. It consists of the future Epics intended to address user needs and deliver business benefits. It also includes architectural Epics required to provide the infrastructure and platform needed to support the realization of business epics.

3.3.2 PROGRAM BACKLOG

The Program backlog is aligned with the Program level of the enterprise and is the single definitive repository for all upcoming work anticipated to advance the ART. It consists primarily of the future Features intended to address user needs and deliver business benefits. It also includes architectural Features required to build the architectural runway.

3.3.3 TEAM BACKLOG

The Team Backlog represents the collection of *all* the things a team needs to do to advance its portion of the system solution. It can contain User Stories, future Features, technical Stories, tasks, defects, infrastructure work, spikes, refactors, and anything else a team needs to do. Stories can be broken down even further into tasks within the Story in order to make the development more manageable. Each Story must also contain a set of acceptance criteria that can be used to ensure that the Story delivers the intended benefits. During Sprint Planning, the team moves items from the Team Backlog into the Sprint Backlog for implementation during the sprint.

3.4 Estimation Techniques

Estimation is an essential part of any project management activity. Agile is no different: We must estimate for scoping, scheduling and budgeting.

The purpose of estimation varies at each level in the IAE Agile Framework. The IAE Framework incorporates several attributes for creating an estimate. Attributes include effort, complexity, risk, team size, domain knowledge, platform quality and technical expertise.

Table 2: Estimation Types

Level	Backlog Items	Measure	Who
Program	Features	T-Shirt Size	Product Management team
Team	User Stories	Story Points	Agile Team members performing the work
Team	Tasks	Hours	Agile Team members performing the work

3.4.1 Release Estimation

The IAE Agile Framework supports each team to independently estimate User Stories given their unique circumstances; however, in order to manage at the Portfolio and Program levels, it is critical to have estimations to forecast and set stakeholder expectations.

The "Velocity" of a team is the number of story points completed in one sprint. Each team will have a different velocity. For consistency at the program level, velocity across the teams is normalized, and program velocity is generated. The program velocity is used to predict releases and is used extensively in release planning.

The following diagram represents the usage of estimates and velocity to determine Release Scope.

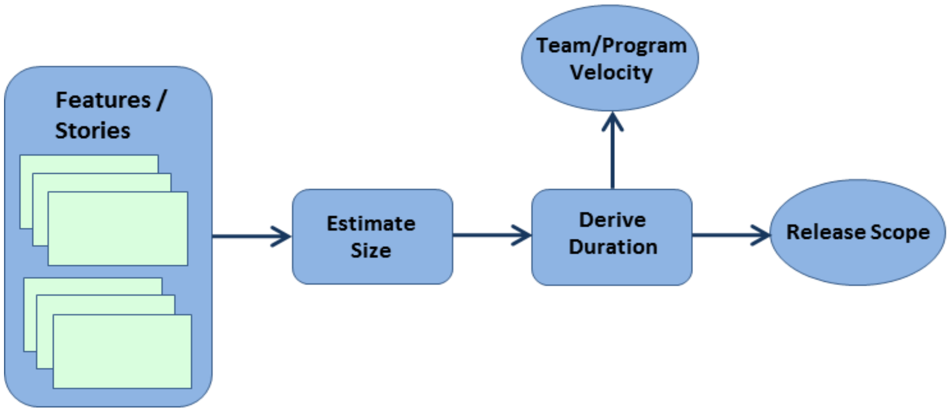
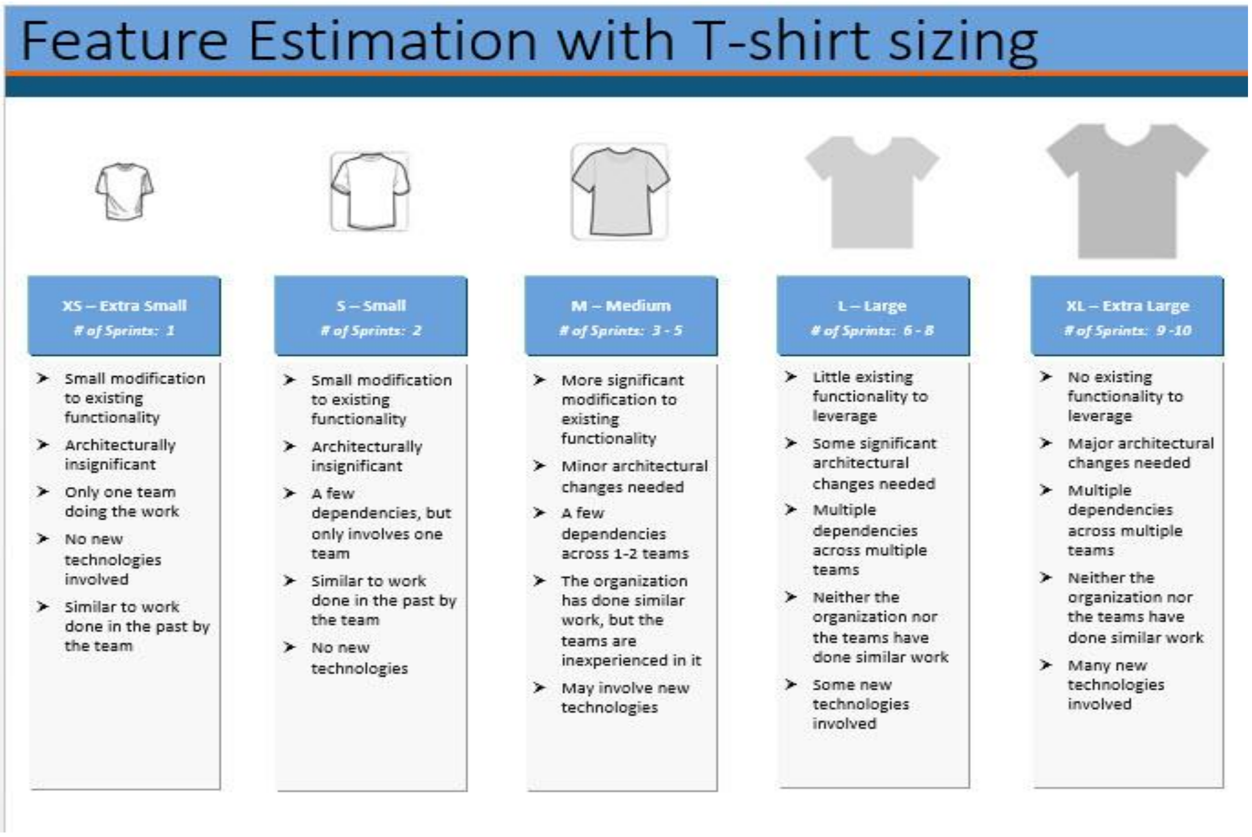


FIGURE 3: RELEASE ESTIMATION FLOWCHART

3.4.2 FEATURE SIZE

At the Feature level, Feature Size, which is the closest proxy for duration, is used for estimation. Feature sizing for each item on the backlog uses the T-shirt sizing method of estimating, which is described more fully in Appendix A3. The figure below provides specific information about how features should be sized.



Feature Estimation with T-shirt sizing

XS – Extra Small *# of Sprints: 1*	S – Small *# of Sprints: 2*	M – Medium *# of Sprints: 3 - 5*	L – Large *# of Sprints: 6 - 8*	XL – Extra Large *# of Sprints: 9 -10*
➢ Small modification to existing functionality ➢ Architecturally insignificant ➢ Only one team doing the work ➢ No new technologies involved ➢ Similar to work done in the past by the team	➢ Small modification to existing functionality ➢ Architecturally insignificant ➢ A few dependencies, but only involves one team ➢ Similar to work done in the past by the team ➢ No new technologies	➢ More significant modification to existing functionality ➢ Minor architectural changes needed ➢ A few dependencies across 1-2 teams ➢ The organization has done similar work, but the teams are inexperienced in it ➢ May involve new technologies	➢ Little existing functionality to leverage ➢ Some significant architectural changes needed ➢ Multiple dependencies across multiple teams ➢ Neither the organization nor the teams have done similar work ➢ Some new technologies involved	➢ No existing functionality to leverage ➢ Major architectural changes needed ➢ Multiple dependencies across multiple teams ➢ Neither the organization nor the teams have done similar work ➢ Many new technologies involved

FIGURE 4: T-SHIRT SIZING FOR FEATURES

The estimates at the Portfolio level support resource allocation and forecasting Epic Completion. The estimates at the Program level support Release Planning-related forecasts. For estimating the scope of a specific release, a more accurate release estimating approach is described below.

3.4.3 Story Points

IAE uses story points to estimate User Stories. Story points are used for relative sizing of user stories. The process of agreeing on a size measurement for the stories or tasks in a product backlog is done by the team responsible for delivering the work, usually using a planning game.

There are several estimation scales such as T-shirt sizes, points, exponential, etc. At IAE, teams will implement a Fibonacci sequence of sorts. Teams will use story point estimation to provide consistency across the program. Story points can be implemented using the planning poker Fibonacci sequence 1, 2, 3, 5, 8, 13, 20, 40 and 100. User Stories with large story point estimates are broken down to ensure that Stories fit within a sprint boundary.

Product Owner(s) create User Stories that represent a wide variety of intent across the ART. These Stories are estimated based on input from each team, creating a set of widely available reference Stories. Each team uses these reference Stories as its baseline for relative estimation of the individual Team backlog.

3.4.4 Task Estimation

Sprint Teams provide Task Estimation in hours for each task. Hours worked on a task are logged each day work is completed. If the original task estimate changes, the team member working on the task provides an updated effort estimate for the remainder of the work. The intent of this estimation is to provide team members with daily feedback on their estimates. It helps improve the team estimation, which ultimately helps with Program and Portfolio estimation.

3.5 Prioritization Techniques

The IAE Agile Framework uses two prioritization techniques: WSJF and the MoSCoW model, which derives its name from "**M**ust have, **S**hould have, **C**ould have and **W**on't have." The following table represents usage of these techniques.

TABLE 3: PRIORITIZATION TECHNIQUES

Level	Artifact	Technique	Responsible
Portfolio	Portfolio Backlog	WSJF	Portfolio Management Team
Program	Program Backlog	WSJF	Product Management Team
Team	Team/Sprint Backlog	MoSCoW	Product Owner(s) and the Agile Team

3.5.1 WEIGHTED SHORTEST JOB FIRST (WSJF)

The WSJF technique helps to prioritize the backlog with an economic view. It is calculated as the Cost of Delay (CoD) divided by job duration. Jobs that can deliver the most value (CoD) and are of the shortest duration are selected first for implementation. The following factors contribute to CoD:

- User-Business Value
- Time Criticality

Cost of Delay = User Business Value + Time Criticality

WSJF = Cost of Delay / Job Size

To calculate WSJF, the stakeholders for each backlog rate each backlog item using a Fibonacci sequence (1, 2, 3, 5, 8, 13, and 20) relative to one another for each of the CoD variables. The emphasis is on the discussion to support prioritization of each backlog item. The item with the highest WSJF also has the highest priority. The owner of a specific backlog facilitates the discussion.

Benefit: WSJF is a technique that prioritizes the most valuable features to the business that can be delivered at the earliest.

3.5.2 MOSCOW MODEL

It is essential that the Team Backlog and the Sprint Backlog are prioritized to enable Agile teams to deliver on the highest value requirements first and deliver business value at the earliest. Once there is a clear set of User Stories, it is important to ensure they are ranked. This helps everyone (customer, designer, developers and testers) understand the most important requirements, in the correct order to develop them, and to understand those that won't be delivered if there are time or resource constraints.

The MoSCoW model is widely used at the Team Level for Team and Sprint Backlog Prioritization.

TABLE 4: MOSCOW MODEL

Letter	Term	Description
M	Must	Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.
S	Should	Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one that can be satisfied in other ways if strictly necessary.
C	Could	Describes a requirement that is considered desirable but not necessary. This will be included if time and resources permit.
W	Won't	Represents a requirement that stakeholders have agreed will not be implemented in a given release, but might be considered for the future.

Benefit: MoSCoW has been used in time-boxed iterative development since the 1980s and is a proven prioritization technique to sort Features (or User Stories) into priority order – a way to help teams quickly understand the customer's view of what is essential for launch and what is not.

3.6 AGILE METRICS

The Agile Manifesto emphasizes the importance of satisfying the customer through early and continuous software delivery. According to an Agile Manifesto principle, the primary measure of progress is working software. IAE encourages all managers and teams to hold their measurements against this critical principle. IAE is committed to showing progress through the demonstration of working software at the end of each sprint and release. In addition to system- and team-level demonstrations, the Agile metrics listed below can help managers and teams monitor progress and focus on quality.

Refer to the Attachment, IAE Agile Metrics (listed in the front matter of this document) for a more detailed description of metrics.

Table 5: Agile Metrics Summary

Metric	Description	Benefit	Usage Level	Owner
Portfolio Progress Report	Roll-up of business and architectural Epics across projects in a portfolio at a glance	Provides a high-level overview of all Epics for monitoring progress.	Portfolio (Team)	Program Managers
Epic Progress Report	The Epic Progress Report shows a list of complete, incomplete and un-estimated stories in an Epic.	Provides ability to track progress of Epics of incomplete or un-estimated work.	Portfolio (Team)	Program Managers
Release Burndown by Iteration	The Release Burndown by iteration chart determines the overall status of the release by plotting Story completion status on a sprint-by-sprint basis.	Provides the current rate at which work is being completed at the Release level.	Program (Team)	Program Managers
Feature Progress Report	Provides business stakeholders with insights into the progress of Features.	Provides the current rate at which work is being completed at the Feature level.	Program (Team)	Program Managers
Sprint Burndown	This is a graphical representation of the actual work completed and work remaining, measured against an idealized line.	Provides the current rate at which work is being completed at the sprint level.	Project (Team)	Scrum Master

Additional Metrics:

Table 6: Portfolio Metrics

Category	Portfolio Metrics	Year 1	Year 2	Year 3	Year 4
Value Delivery	# of Releases to Production				
Cost Savings	# of Legacy Systems Retired				
Quality	# of Defects				
Customer	% of Customer Satisfaction				

Table 7: Program Metrics

Program performance Metrics	Release 1	Release 2	Release 3	Release 4
Release Velocity				
# of Features Planned				
# of Features Accepted				

PROGRAM PERFORMANCE METRICS	RELEASE 1	RELEASE 2	RELEASE 3	RELEASE 4
% of Features that are Architectural				
# of Stories Planned				
# of Stories Accepted				
QUALITY METRICS				
% of Unit Test Coverage				
% of Tests Automated				
# of System Tests				
# of Defects Outstanding+				

TABLE 8: TEAM METRICS

TEAM PERFORMANCE METRICS	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4
Velocity Planned				
Velocity Actual				
# of Stories Planned				
# of Stories Accepted				
% of Stories Accepted				
QUALITY METRICS				
% of Unit Test Coverage				
% of Test Automated				
Total # of Tests				
# of Builds per Day				
Average Build Duration				
# of Defects Outstanding				

3.7 AGILE TOOLS

IAE has adopted a suite of automated tools to implement the Agile Life Cycle Management. JIRA is being used as IAE's Backlog Management tool. The Atlassian suite contains JIRA and Confluence, which provide essential features for implementing the Agile process described in this framework. Additional plug-in software packages have been implemented to enhance the suite and provide an enterprise capability.

4 Introduction to the Agile Process

4.1 Framework Levels

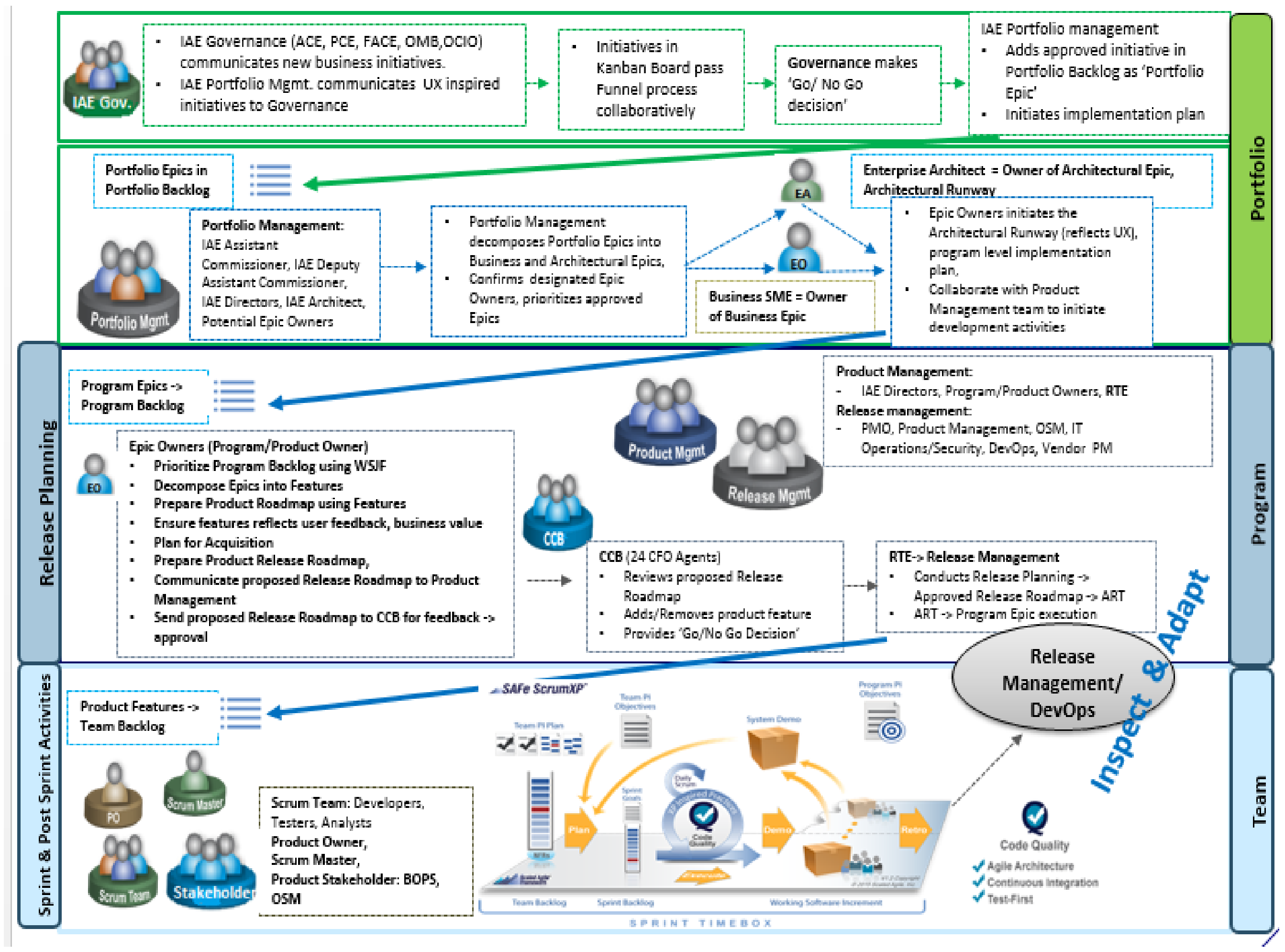
The IAE Agile software development framework is designed to enable agility at scale across the enterprise. The process used to implement this methodology is applied on all three levels to enable the scaling of Agile across the organization. This framework establishes the platform where Agile Teams work in a synchronized cadence with one another to create integrated software that satisfies organizational goals.

The following table summarizes the three levels of the IAE Agile Framework:

Table 9: IAE Agile Framework Levels and Governance Framework

Level	Purpose	Alignment
Portfolio	• Define new business initiatives, strategic themes, and Epics • Define and prioritize the Portfolio Backlog • Allocate resources, budget and capacity • Move Epics into implementation • Guide program execution • Provide overall governance	Align portfolio to the enterprise strategy, goals and objectives
Program	• Provide program vision and roadmap • Define and prioritize the Program Backlog • Implement Releases via prioritized Features • Establish cadence and synchronization across the program • Provide overall Release planning, execution, and management • Enable transparency and collaboration across the program • Define Program Metrics • Demonstrate and deliver an integrated system	Align program implementation to the portfolio vision
Team	• Deliver to Release Objectives • Build high quality software • Maintain continuous coordination and collaboration across teams • Provide frequent demonstration of working software • Incorporate improvement continuously	Align product development and delivery to the program vision

The following diagram illustrates the IAE High Level Process Flow.



- IAE Governance (ACE, PCE, FACE, OMB,OCIO) communicates new business initiatives.
- IAE Portfolio Mgmt. communicates UX inspired initiatives to Governance

- Initiatives in Kanban Board pass Funnel process collaboratively

Governance makes 'Go/ No Go decision'

IAE Portfolio management
- Adds approved initiative in Portfolio Backlog as 'Portfolio Epic'
- Initiates implementation plan

Portfolio Epics in Portfolio Backlog

Portfolio Management: IAE Assistant Commissioner, IAE Deputy Assistant Commissioner, IAE Directors, IAE Architect, Potential Epic Owners

- Portfolio Management decomposes Portfolio Epics into Business and Architectural Epics,
- Confirms designated Epic Owners, prioritizes approved Epics

Enterprise Architect = Owner of Architectural Epic, Architectural Runway

Business SME = Owner of Business Epic

- Epic Owners initiates the Architectural Runway (reflects UX), program level implementation plan,
- Collaborate with Product Management team to initiate development activities

Program Epics -> Program Backlog

Product Management:
- IAE Directors, Program/Product Owners, RTE

Release management:
- PMO, Product Management, OSM, IT Operations/Security, DevOps, Vendor PM

Epic Owners (Program/Product Owner)
- Prioritize Program Backlog using WSJF
- Decompose Epics into Features
- Prepare Product Roadmap using Features
- Ensure features reflects user feedback, business value
- Plan for Acquisition
- Prepare Product Release Roadmap,
- Communicate proposed Release Roadmap to Product Management
- Send proposed Release Roadmap to CCB for feedback -> approval

CCB (24 CFO Agents)
- Reviews proposed Release Roadmap
- Adds/Removes product feature
- Provides 'Go/No Go Decision'

RTE-> Release Management
- Conducts Release Planning -> Approved Release Roadmap -> ART
- ART -> Program Epic execution

Product Features -> Team Backlog

Scrum Team: Developers, Testers, Analysts
Product Owner,
Scrum Master,
Product Stakeholder: BOPS, OSM

FIGURE 5: IAE AGILE FRAMEWORK HIGH LEVEL PROCESS

5 Portfolio Level Process Overview

At this level, the Portfolio Management Team reviews all strategic initiatives and business and technology (architecture) Epics to ensure that they align with the overall enterprise strategy, to provide the greatest benefit and value for the organization and its key stakeholders. A comprehensive Portfolio Backlog is created at this level with the input of the stakeholders. The portfolio backlog is created first then prioritized, and Epics are created after an initial analysis. The final list of Epics is prioritized, and budget and resources are allocated to the approved initiatives (Epics). The ART formation is initiated at this level, and Epics are allocated to the ART.

Refer to Appendix B for Portfolio Level Details.

The following diagram represents the Portfolio Level Process, which is at the highest level in the framework.

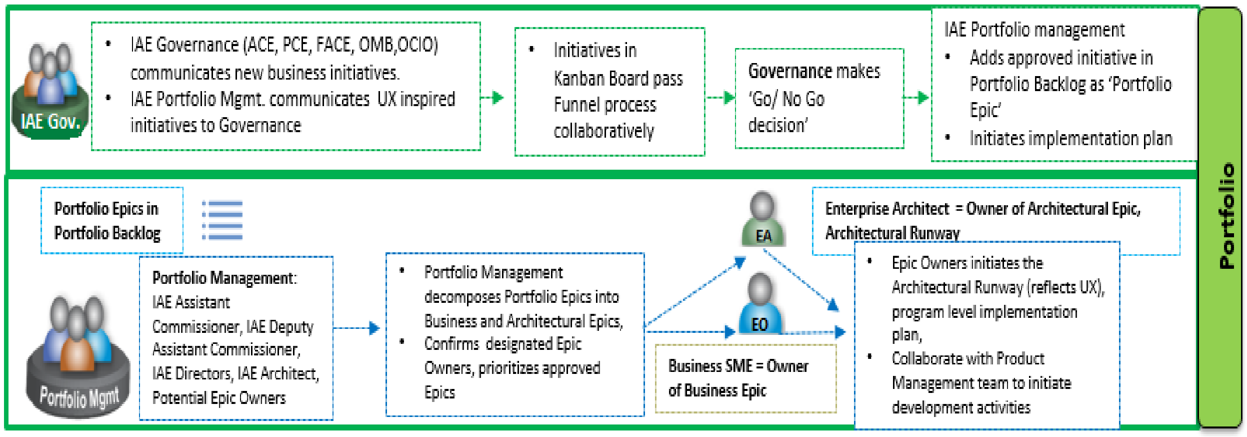

IAE Gov.
IAE Governance (ACE, PCE, FACE, OMB,OCIO) communicates new business initiatives.
IAE Portfolio Mgmt. communicates UX inspired initiatives to Governance
Initiatives in Kanban Board pass Funnel process collaboratively
Governance makes 'Go/ No Go decision'
IAE Portfolio management
Adds approved initiative in Portfolio Backlog as 'Portfolio Epic'
Initiates implementation plan
Portfolio Epics in Portfolio Backlog
Portfolio Mgmt
Portfolio Management: IAE Assistant Commissioner, IAE Deputy Assistant Commissioner, IAE Directors, IAE Architect, Potential Epic Owners
Portfolio Management decomposes Portfolio Epics into Business and Architectural Epics,
Confirms designated Epic Owners, prioritizes approved Epics
EA
EO
Enterprise Architect = Owner of Architectural Epic, Architectural Runway
Business SME = Owner of Business Epic
Epic Owners initiates the Architectural Runway (reflects UX), program level implementation plan,
Collaborate with Product Management team to initiate development activities
Portfolio

Figure 6: Portfolio Level Process

The following sections provide a high level overview of the Portfolio Level processes:

5.1 Governing Body Communicates New Initiatives

The Governing Body communicates enterprise vision and strategy. It defines strategic themes and new business initiatives that influence Epics evaluation, program vision and the roadmap. The Governing Body meets at the regular ACE/FACE/PCE meetings (Award Committee for E-Government / Financial Assistance Committee for E-Government / Procurement Committee for E-Government), which take place quarterly.

5.2 Portfolio Backlog Formation

The Portfolio Backlog is initially formed by adding potential Business and Architectural Epics to the Kanban. These potential epics arise when “big business and architectural ideas” are introduced, also known as the Problem/Solution Needs Identification, from specific, itemized business objectives to the evolving enterprise business strategy.

5.3 Epic Understanding and Alternatives

The purpose of the Epic Kanban Process is to provide visibility and guide new initiatives through the analysis process, to the Go/No Go decision. The Product Management Team reviews and analyzes new initiatives, develops Epic value statements, and develops lightweight business cases. They prepare recommendations of proposed Epics for final approval by the Portfolio Management Team or the Change Control Board (CCB) Team. If the Epic requires review by the CCB, it is reviewed by them before it’s moved to the Portfolio Backlog. Otherwise, once the Portfolio Management Team approves an Epic, it is added to the Portfolio Backlog.

5.4 Prioritization of the Portfolio Backlog

The Portfolio Management Team is responsible for managing investments, driving Epic development, monitoring Epic implementation progress, and grooming the Portfolio Backlog. During the Portfolio Management meeting, Epics in the Portfolio Backlog are reviewed on a periodic cadence and prioritized using WSJF based on the overall enterprise vision and strategic direction. The prioritization of Epics ensures that the highest priority Epics are promoted to implementation when there is sufficient capacity from the ARTs. The Portfolio Management Team provides direction and guidance on the overall solution strategy for the implementation of Portfolio Epics and the execution of releases.

5.5 Portfolio Monitoring

The Portfolio Management Team performs a Portfolio Health Check, reviewing overall status, dashboards and metrics on the implementation Portfolio Epics and the execution of Releases. The Portfolio Management Team takes action to help resolve any impediments escalated from Releases and Teams. Corrective actions are applied to ensure continuous delivery of value.

6 Program Level Process Overview

At this level the Program Epics are reviewed and decomposed by the Product Management Team, which creates the Release Roadmap using techniques such as Minimum Viable Product, also known as the product with the highest return on investment versus risk. The Features that are decomposed from the Program Epics are then prioritized using techniques such as WSJF.

All the teams working in an ART are synchronized, i.e., they start and stop at the same time. The cadence of delivery is also orchestrated by the RTE, a member of the Release Management team. Program Risk Management occurs at this level, and risks are escalated from Team to Program to Portfolio level.

Refer to Appendix C for Program Level Details.

The following diagram represents the Program Level Process, which is at the midlevel in the framework.

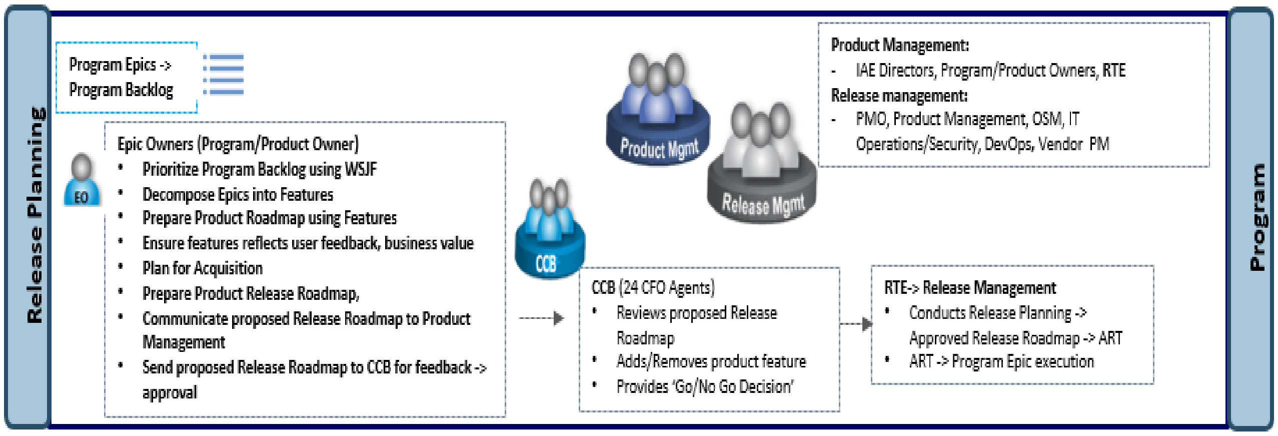

Release Planning
Program
Program Epics -> Program Backlog
Epic Owners (Program/Product Owner)
• Prioritize Program Backlog using WSJF
• Decompose Epics into Features
• Prepare Product Roadmap using Features
• Ensure features reflects user feedback, business value
• Plan for Acquisition
• Prepare Product Release Roadmap,
• Communicate proposed Release Roadmap to Product Management
• Send proposed Release Roadmap to CCB for feedback -> approval
EO
CCB
Product Mgmt
Release Mgmt
Product Management:
- IAE Directors, Program/Product Owners, RTE
Release management:
- PMO, Product Management, OSM, IT Operations/Security, DevOps, Vendor PM
CCB (24 CFO Agents)
• Reviews proposed Release Roadmap
• Adds/Removes product feature
• Provides 'Go/No Go Decision'
RTE-> Release Management
• Conducts Release Planning -> Approved Release Roadmap -> ART
• ART -> Program Epic execution

Figure 7: Program Level Process Overview

At the Program level, the ART teams take responsibility for implementing Epics, Features and User Stories. All of the ARTs running within the enterprise work in a synchronized cadence. Coordinating efforts among multiple teams in regular, repeatable, short intervals leads to decreased variability in the work and results in a more efficient, dependable, reliable and adaptable work product.

The following sections provide a high level overview of the Program Level Processes:

6.1 Program Backlog Grooming

Epics at the Program level are decomposed into Features and sub-Features in the Program Backlog during the Backlog Grooming process. This is done primarily by the Product Management Team, Epic Owners and the System Architect, but also with assistance from the System Team, Release Management Team, the Development Operations (DevOps) Team, and the Independent Validation & Verification (IV&V) Team.

The Backlog Grooming is used to prepare for the Release Planning ceremony. During Backlog Grooming, existing Features are reviewed and elaborated upon, which includes defining acceptance criteria and establishing estimates. Larger Features are evaluated for breaking down into smaller Features or sub-Features. Features are also evaluated to determine if they need to be decomposed into related architectural Features. In addition, the capacity that can be allocated for these architectural Features in upcoming releases is determined. Finally, the Features and sub-Features in the Program Backlog are prioritized using the WSJF approach.

6.2 Change Control Board (CCB) Approval

The CCB serves as the content authority that is empowered to decide what gets implemented on the ART. It reviews, edits and approves Features to be implemented in the upcoming release via the approval of the Release Roadmap.

6.3 Release Planning

Agile releases run on a three-month cadence of Release Planning -> Implementation -> Release Demo -> Release Retrospective. Release Planning is the ceremony that ensures the program stays in this rhythm of continuous development and delivery. Release Planning takes place at the beginning of each release. This ceremony is facilitated by the RTE and supported by all members of the program. This face-to-face ceremony level sets everyone's understanding of the program's vision and roadmap and secures the development teams' commitment to the objectives for the upcoming release. The development teams will assess and mitigate technical dependencies across teams. This high level of collaboration ensures the resulting plan is realistic and implementable.

6.4 Scrum of Scrums and Product Owner Scrum of Scrums

The Scrum of Scrums is necessary to manage dependencies across teams during development of the release and to manage risk. The RTE, also known as the Chief Scrum Master, facilitates the Scrum of Scrums meeting with all the teams' Scrum Masters and the Program Manager. The Scrum of Scrums ensures that teams are meeting their sprint objectives, removes any problems that are causing delays or confusion, and raises the level of awareness of dependencies among the Agile teams. Similarly, a Product Owner Scrum of Scrums is used to promote collaboration among Product Owners and provides an opportunity for Product Owners to discuss the interdependencies among their program areas. The RTE facilitates this meeting where Product Owners report status, planned work and current impediments to progress.

6.5 Release Management and Release Demo

One of the Agile Manifesto principles states that the primary measure of progress is working software. The Release Management process monitors the progress of sprints toward the realization of important features. In some cases, no software will be released until the end of the quarterly release cycle; however, to reap the benefits of the Agile development process, it might be desirable to have additional releases of working software within the quarterly cycle. This could be required to support an external milestone, to promote bug fixes, or just to provide benefit to users at the earliest point in time.

The IAE Agile Framework prescribes a Release Demo prior to each release (major or minor) as well as at the end of each sprint, when Features are completed and can be presented via working software. The Release Demo provides an integrated, program-level view of all new Features delivered by all the teams in the most recent iteration. The Release Demo lets the teams showcase their accomplishments and allows business stakeholders and customers to provide immediate feedback. The Release Demo also ensures that full and continuous integration takes place frequently. The System Team stages the Release Demo with support from individual development teams. At the end of a release, a retrospective is held immediately after the Release Demo to discuss how future releases can be executed more effectively.

6.6 Release Retrospective

A principle in the Agile Manifesto states that at regular intervals, the team reflects on how to become more effective. IAE adheres to this principle by holding a Release Retrospective, also known as the Inspect and Adapt workshop, at the end of each release. During this ceremony, participants review the agreed upon quantitative metrics to evaluate the performance of the current release. This is followed by a retrospective on the release and a problem-solving workshop. Corrective action plans are devised and reviewed by the group. As a result of this workshop, teams come up with a set of improvement User Stories that are added to the program backlog and incorporated into the next release planning session, thus assuring that action will be taken.

Benefits of Retrospective:

- Incorporates timely feedback on what works and what doesn’t.
- Provides early and frequent feedback to adapt.
- Boosts team morale with collaborative processes.
- Generates new ideas for improvements.
- Keeps the focus on the identified goals of the sprint, release and project.
- Celebrates successes!
- **Improved productivity**: By applying lessons learned and reducing rework, the team can get more productive work done.
- **Improved capacity**: Retrospectives provide a venue for spreading knowledge, and as the number of people who have the knowledge increases, so does the number of people who can perform tasks associated with the knowledge.
- **Improved quality**: We can improve quality on our projects by finding the circumstances that led to defects and removing the causes.
- **Improved capacity**: Retrospectives focus on finding process efficiency improvements, which can improve teams’ capacity to do the work.

7 Team Level Process Overview

The team level in the IAE Agile Framework consists of teams who are empowered to make local decisions and are accountable to deliver on the Program Commitments. All the teams develop in cadence and synchronization to the Release Schedule.

Refer to Appendix D, Team Level Details.

The following diagram represents the Team Level Process, which is at the lowest level in the framework.

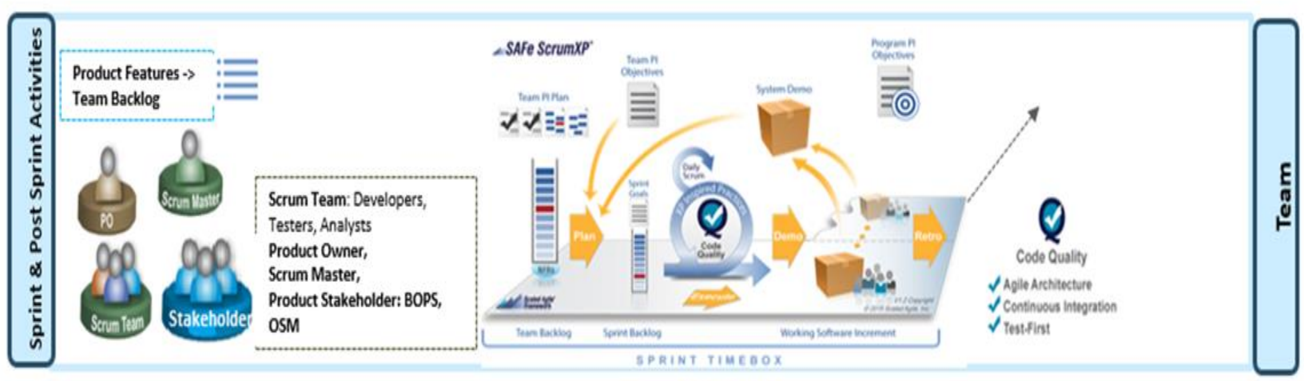


Figure 8: Team Level Process Overview

The primary purposes of the team process are to ensure that quality software is developed and that the product aligns with the program vision.

The following sections provide a high level overview of the Team Level Processes.

7.1 Team Backlog Grooming

Team Backlog Grooming ensures that the priorities align with the latest program direction and that the backlog is elaborated sufficiently to support successful Sprint Planning. The Agile Team is expected to support the Product Owner in grooming and refining the Team Backlog. Technical expertise will help establish accurate size estimates for items in the backlog. Large user stories may be decomposed into smaller stories and tasks.

7.2 Sprint Planning

During Sprint Planning, each team picks high-priority user stories from the Team Backlog and commits them to the Sprint backlog for execution. The team's backlog should be set up by the Product Owner prior to Sprint Planning. Sprint Planning is typically time-boxed to four hours or fewer. The output of this process is the team's commitment to implement stories in the Sprint backlog. The team also commits to achieve Sprint goals that support the current release objectives.

7.3 Daily Scrum

The Daily Scrum is designed to help the team set the context for the coming day's work and to communicate the progress of the work completed from the previous day. This meeting is time-boxed to 15 minutes to keep the discussions concise and relevant. The team can quickly share information on what was accomplished yesterday, what is planned for today, and report any impediments. The Scrum Master is responsible for helping the team resolve impediments.

7.4 Sprint Demo

The purpose of the Sprint Demo is to show the progress the team has made to the Product Owner and other stakeholders. The Sprint Demo, sometimes called the Sprint Review or Team Demo, takes place at the end of each two-week Sprint. The Sprint Demo starts with a quick review of the Sprint goals and metrics, followed by a demonstration of each completed Story. After the demonstration, the team discusses which Stories were not completed and why they were not done. A retrospective is held immediately after the Sprint Demo to discuss how the team can be more effective going forward.

7.5 Sprint Retrospective

At the end of each Sprint, the team holds a retrospective to reflect on how to become more effective. This meeting is typically time-boxed to an hour for team members to discuss what went well, what didn't, and what the team can do better next time. The team reviews performance metrics and discusses any impediments and challenges faced during the past iteration. Root cause analysis is performed, and corrective actions are logged as User Stories in the Team Backlog. The team picks one or two improvement items to target for the next Sprint. If necessary, the improvement items may become backlog items in the form of User Stories that are prioritized by the Product Owner with input from the team.

A. General Information

Appendix A1: Requirements Model Details

Epics

Epics are enterprise initiatives that are large enough such that their development could span multiple releases and could impact multiple release trains. There are business Epics (customer-facing) and architectural Epics (technology solutions). It is also possible for Epics to originate from the program level.

Because of their size and impact on the organization, Epics must go through a careful evaluation process to determine their viability. The evaluation system used is called the Kanban system. Two criteria used in the evaluation of Epics are the Epic Value Statement ***(refer to Appendix A2: Epic Value Statement Template)*** and the Epic Lightweight Business Case. The Epic Value Statement is used in the Kanban Review step and is intended to provide just enough information to have a meaningful discussion about the proposed initiative.

Features and Sub-Features

Epics decompose into Features, the next smallest requirement artifact. They are maintained in the Program Backlog and are sized to fit within one release. They can originate at the Program level, or they can derive from Epics defined at the Portfolio and Program levels. IAE has added a level below Feature, called Sub-Feature, which is simply a way of breaking down Features into smaller pieces of work. Features may or may not be broken down into Sub-Features.

A Features and Benefits Matrix (FAB) can be used to describe each Feature. A Feature contains three parts. The first part, Feature, contains a short phrase that gives the Feature a name and some implied context. The recommended format of this short phrase is an action verb followed by the scope or capability of the Feature. The second part, Benefit, provides a short description of the benefit of the Feature to the user and the organization. The last part, Acceptance Criteria, is used to determine that the Feature has been implemented correctly. Acceptance criteria help set the scope of the Feature and support development of associated User Stories and functional tests.

User Stories

Features/Sub-Features decompose into User Stories, which should be constructed in such a way that they can be completed within one Sprint. User Stories describe the detailed implementation work and are the primary element of the team backlog. In addition to being derived from Features, Stories can also originate at the team level.

In software development and product management, a User Story is a description that captures what a user does or needs to do as part of his or her job function. User stories are used as the basis for defining the functions a business system must provide. They capture the 'who', 'what' and 'why' of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a small paper notecard.

User stories typically follow the pattern “As a *<persona>*, I can *<activity>* so that *<business value>*.” If the “persona” is a device or another system, simply substitute the device or system name in place of the persona. Technical Stories describe other types of necessary system behavior. Technical Stories can be expressed in technical, rather than user-centric language, but the “…so that…” portion of the story should be retained so that the motivation for the story is understood.

User Stories are prioritized by the product owner. User stories can be broken down into tasks and estimated by the developers. User stories are accepted by the product owner based on the predefined Acceptance Criteria.

Personas

Personas are people or system functions that are described in terms of category of person/system that interacts with the software product (CFDA, WDOL, etc.). As a software product is generally intended for use by more than one category of person, with potentially different preferences and expectations of the product, the team creates one persona for each category it deems important to serve.

Different Personas are described in User Stories so developers can understand what privileges (authentication and authorization) need to be developed for each Persona.

Testers test using different user credentials to mirror the functionality based on the category of person/system (AKA, Persona).

Acceptance Criteria

Acceptance Criteria are a set of statements, each with a clear pass/fail result, that specify both functional (e.g., minimum viable product) and non-functional (e.g., compliance to standards) requirements applicable at the current stage of project integration. These requirements represent “conditions of satisfaction.” There is no partial acceptance: Either a criterion is met or it is not.

- **Functional Criteria:** Identify specific user tasks, functions or business processes that must be in place. A functional criterion might be “A user is able to access a list of available reports.”
- **Non-functional Criteria:** Identify specific non-functional conditions the implementation must meet, such as design elements. A non-functional criterion might be “Workflow buttons comply with the Human Interface Guidelines.”
- **Performance Criteria:** If specific performance is critical to the acceptance of a User Story, it should be included. This is often measured as a response time, and it should be spelled out as a threshold such as “less than 2 seconds response time.”

Acceptance Criteria must be:

- Expressed clearly

- In simple language the customer would use, without ambiguity as to what the expected outcome is
- Actionable
- Comprehensive in scope, i.e., it must include what is acceptable and what is not acceptable
- Testable, i.e., easily translated into one or more manual/automated test cases.

Acceptance Criteria define what must be done in order for a Feature or User Story to be accepted by the product owner.

Acceptance Criteria may reference what is in the project's other User Stories or design documents to provide details, but they should not be a re-hash of them.

Acceptance Criteria should be relatively high-level while still providing enough detail to be useful.

Acceptance Criteria should state intent but not a solution (e.g., "A manager can approve or disapprove an audit form" rather than "A manager can click an 'Approve/Disapprove' radio button to approve an audit form"). The criteria should be independent of the implementation. Ideally, the phrasing should be the same regardless of target platform.

Acceptance Criteria are required for all Features and User Stories.

Acceptance Criteria must be defined before Features or User Stories are worked by the implementation/development team.

Acceptance Criteria should be complete before Release Planning and Sprint Planning.

Acceptance Criteria are documented in JIRA:

- Acceptance Criteria are stored in the program/team backlog
- Acceptance Criteria are documented in the description field of a Feature below the benefit statement
- Acceptance Criteria are documented in the description field of a User Story below the persona description

Examples of Acceptance Criteria:

Feature: *Developers can use published API's*

Acceptance Criteria:

- ✓ Developer is able to invoke a published API
- ✓ Developer is notified of invalid key usage
- ✓ IAM access-based roles are enforced

User Story: *As an Administrator, I want to be able to create User Accounts so that I can grant users access to the system.*

Acceptance Criteria:

- ✓ As an Administrator, I can create User Accounts.
- ✓ I can create a User Account by entering the following information about the User: a. Name; b. E-mail address; c. Phone Number; d. License Number

(Power/Basic/None); e. Account Status (Active/Inactive); f. Reports to (from a list of "Active" Users)

- ✓ I cannot assign a new User to report to an "Inactive" User
- ✓ I cannot assign a new User to report to a User if it creates a cyclical relationship (e.g., User 1 reports to User 2, who reports to User 1)
- ✓ The system notifies me that it sent an e-mail to the new User's e-mail address, containing a system-generated initial password and instructions for the person to log in and change the password.
- ✓ I am able to verify with the intended recipient of the e-mail that it was received.

User Story: *As a Level 0 Admin, I want to view a list of departments so that I can assess my realm of responsibility.*

Acceptance Criteria

- ✓ I can see a list of departments that I am authorized to see
- ✓ I can see summary department information to include name and code

User Story: *As Level 1 Admin, I can see information about the department I am assigned to.*

Acceptance Criteria (using Behavior-Driven Development (BDD) format)

- ✓ AC1:
  Given I am an Level 1 Admin
  When I am on FH home page
  Then I see my department information: department name, code
- ✓ AC2:
  Given I am an Level 1 Admin
  And I am on FH home page
  When I select my department name
  Then I see my department detail information: department name, code, point of contact (name, e-mail, phone number)

Splitting Epics, Features, and User Stories

(from Scaled Agile Framework; Humanizing Work, 2009)

Epics must be split into Features to drive actual implementation. Features and User Stories may need to be decomposed into smaller parts to in order to be implemented within a release or a Sprint. Below are methods for splitting Epics, Features and User Stories, along with examples:

1. **Product/Subsystem/Component**: Epics often affect multiple products, subsystems or large components. In such cases, splitting by these aspects can be an effective implementation technique.

   Example: Allow multiple user profiles

   a. Allow multiple profiles in the external website.
   b. Allow multiple profiles in the admin system.

2. **Success Criteria**: The Epic's success criteria often provide hints as to how to incrementally achieve the anticipated business value.

   Example: Implement new location artifacts in search results. Success Criteria: a) Locations should provide additional filtering methods when other disambiguation methods aren't useful; b) Provide detailed location of a person.

   a. Provide state information in the search.
   b. Implement compound location: state and city.

3. **Major Effort First**: Sometimes a requirement can be split into several parts, where most of the effort will go toward implementing the first one.

   Example: Provide room reservation system.

   a. Add ability to reserve recurring reservations.
   b. Implement multi-location reservation capability.

4. **Simple/Complex**: Capture the simplest version of the requirement and then add additional requirements for all the variations and complexities.

   Example: Implement Single Sign On (SSO) across all products in the suite

   a. Implement SSO management capability in our simplest product
   b. Implement SSO in our most complex product

5. **Variations in Data**: Data variations and data sources are another source of scope, complexity and implementation management.

   Example: Internationalize all end-user facing Web pages.

   a. Implement in Spanish.
   b. Implement in Japanese.
   c. Prioritize the rest by current market share.

6. **Market Segment/Customer/Class of User**: Segmenting the market or customer base is another way to split a requirement. Address the one that has a higher business impact first.

   Example: Implement customer feedback opt-in functionality.

   a. Implement for current partners.
   b. Implement for all major marketers.

7. **Nonfunctional Requirements (NFRs)**: Split Epics by implementing the initial capability first and then incrementally injecting improvements in system qualities (NFRs such as security, reliability, maintainability, scalability and usability).

   Example: Provide access to vendor marketplace

   a. Access is provided to all registered users.
   b. Vendor marketplace can be accessed within 3 seconds.
   c. Vendor profile data can only be accessed by account owner.

8. **Risk Reduction/Opportunity Enablement**: Given their scope, Epics can be inherently risky. Use risk analysis and conduct the riskiest parts first.

Example: Implement filtering of search results by complex user-defined expressions.

a. Implement negative filtering.
b. Implement filtering expressions with logical operations.

9. **Use Case Scenarios**: Use cases can be used in Agile to capture complex user-to-system or system-to-system interaction. Split according to the specific scenarios or user goals of the use case.

   Example: People search functionality.

   a. Goal 1: Find connection to a person.
   b. Goal 2: Find connection to a company.
   c. Goal 3: Distinguish storing and weak connections.

10. **Workflow Steps**: Features/User Stories may have multiple steps in order to complete a Web feature on the website, such as create content, review content, approve content and publish content.

    Example: As a content manager, I can publish a news story to the corporate Website.

    a. I can publish a news story directly to the corporate Website.
    b. I can publish a news story with editor review.
    c. I can publish a news story with legal review.

11. **Business Rule Variations:** Features/User Stories may require different business rules that will be implemented.

    Example: As a user, I can search for flights with flexible dates.

    a. n days between x and y
    b. a weekend in December
    c. ± n days of x and y

12. **Data Entry Methods:** Features/User Stories implement different ways to enter data.

    Example: As a user, I can search for flights between two destinations.

    a. Using simple date input
    b. Using a fancy calendar UI

13. **Defer Performance:** Implement different levels of performance

    Example: As a user, I can search for flights between two destinations

    a. Slow – just get it done, showing a "searching" animation
    b. In under 5 seconds

14. **Operations (e.g. CRUD):** Separate create, read only, update, and delete operations

    Example: As a user, I can manage my account.

    a. I can sign up for an account
    b. I can edit my account settings
    c. I can cancel my account

15. **Break Out a Spike –** Conduct a spike to help understand the technical feasibility before implementation.

    Example: As a user, I can pay by credit card.

    a. Investigate credit card processing
    b. Implement credit card processing (as one or more stories)

Definition of Ready

Definition of Ready (DoR) is a set of agreements that lets everyone know when Release Planning can be started, when a Feature is ready to be implemented, when Sprint Planning can be started, and when a User Story is ready to be implemented by the development team.

The figure below lists Definition of Ready for Release Planning, Features, Sprint Planning, and User Stories. These definitions can be tailored during the Release and Sprint Planning ceremonies.

Table 10: Definition of Ready

Definition Of Ready	
User Story	A User Story is ready when: • The business value is articulated using the user story format • It meets INVEST (is Independent, Negotiable, Valuable, Estimable, Small and Testable Guidelines) • It is small enough to finish within a sprint • The description includes: ○ Acceptance criteria ▪ describes/defines when it is done ○ Performance criteria ▪ e.g., users/load, response time, retention, etc. ○ Other DoD criteria ▪ e.g., reviews, artifacts, compliance/regulatory requirements, etc. ○ External dependencies identified (if applicable) ○ External SME identified with contact info e.g., for coordination, reviews, discussions • The story is prioritized • Other items are known &/or ready enough for building to begin: ○ Design artifacts ▪ e.g., UI, DM/DB, etc. ○ Infrastructure ▪ e.g., dev/QA/integration/staging/prod environment, continuous integration, permissions, etc. • The team understands the intent, and knows how to demo the story • The person accepting the story is identified

DEFINITION OF READY	
Feature	A Feature is ready when: • The Business value is articulated • Feature can fit in a Release • Feature is prioritized • Has Owner identified • The Feature follows the appropriate format: ○ Name: Action verb followed by a short phrase ○ Benefit statement: ▪ Who benefits ▪ What the benefit is ○ Acceptance criteria ▪ describes/defines at a high-level when it is done ○ Other information as applicable: ▪ DoD criteria met ▪ External dependencies identified ▪ External SMEs identified, with contact info • Other items are known and/or ready enough: ○ Architecture artifacts ▪ information architecture, UX, conceptual architecture, etc. ○ Infrastructure ▪ environments, etc. • The team understands the intent • A decomposition strategy has been determined ○ e.g., what splitting pattern is best for breaking it down into smaller items
Release Planning	We are ready for Release Planning when: • The Facility is ready ○ Room reserved and set up ○ Work space/surfaces ○ Supplies (e.g. sticky notes, flip-charts, markers/pens, string, paper roll, etc.) ○ Technology (phone, network/Internet, PCs, etc.) ○ Collaboration tools (video, screens, virtual meeting space, etc.) • The Participants are ready ○ Attendees invited and ready to participate ○ Attendees know and understand their role, and are ready to participate ○ Participants know and understand the expected outcomes of planning • The Content is ready ○ Agenda ○ Business Context ▪ Includes business drivers ○ Product Vision ▪ Includes release objectives

Definition Of Ready	
	▪ presented by product management ○ Architecture Vision ▪ Includes description of the architecture needed to enable the business objectives ○ Release Objectives ○ Roadmap ○ Release Backlog with prioritized Features ○ All Features targeted for the release are ready (i.e., meet DoR for a Feature)
Sprint Planning	We are ready for Sprint Planning when: • Business user story is written using INVEST criteria • Business user story contains acceptance criteria • Sprint backlog is prioritized • The Sprint Backlog contains all potential work for the upcoming sprint ○ i.e., no hidden work ○ items are targeted for the Sprint but not yet committed • Sprint objectives/goals are defined • The team capacity for the Sprint is calculated • All items targeted for the Sprint are ready ○ i.e., meet DoR for a User Story

Definition of Done

The Definition of Done (DoD) defines all steps necessary to deliver a completed product with the best quality possible. DoD is a tool for bringing transparency, related more to a quality of a product than its functionality. DoD requires Agile teams to adhere to a common understanding of completion before moving the work in a potentially releasable state. The figure below lists Definitions of Done at the Release level, Feature level, and User Story levels. These definitions can be tailored during the Release Planning ceremony.

Release

- All features for the releasable set are done and meet acceptance criteria
- End-to-end system integration and system/performance testing done
- Full regression testing done; automated where practical
- No must-fix defects
- User, release, installation documentation complete
- 100% of features are developed with an Infrastructure-as-Code implementation
- 100% of code is open source-able and available in GitHub
- Platform can be built by any third party using IaaS code base
- Feature set accepted by Product Management

Feature

- Features meet acceptance criteria
- All stories for the features are done
- Code deployed to QA/Staging and integration tested
- Functional regression test complete; automated where practical
- Non-Functional Requirements met*
- No must-fix defect
- Documentation related to the Feature has been updated
- Feature demonstrated in Release Demo
- Feature included in build definition and deployment process
- Feature/s accepted by the Product Owner/Program Managers

User Story

- Automated unit test coverage
- Stories satisfy acceptance criteria
- Unit tests and acceptance tests passed
- Cumulative unit tests passed
- NFRs* (508, FISMA, Agile Framework, EA Framework, API standards, Coding standards, UX standards) met
- Code checked in, merged into mainline, and publish to code repository
- Coding Standards followed
- Code peer reviewed
- Code is deployable to a production environment
- Documentation related to the user story has been updated
- No must-fix defect
- Story accepted by product owner

FIGURE 9: DEFINITION OF DONE

Appendix A2: Epic Value Statement

Forward Looking Position Statement	
For	<Customer>
Who	<do something>
The	<solution>
is a	<something – the "how">
That	<provides this value>
Unlike	<competitor, current solution or non-existing solution>
our solution	<does something better – the "why">
Scope	
Success Criteria:	
In Scope:	
Out of Scope:	
NFRs:	

FIGURE 10: EPIC VALUE STATEMENT TEMPLATE

Appendix A3: Estimation Techniques

There are three techniques for estimating Features, each having an increasing level of precision.

- Preliminary Relative Estimate - simply provides an initial, rough estimate. With this technique, use relative estimating to compare the relative size of one Feature to another.
- Gross Absolute Estimate - uses historical comparisons to provide a bit more accuracy. This estimation technique compares new Feature size to the story points required to deliver comparable Features in prior periods.
- Derived Absolute Estimate - most accurate type of estimate where Features are broken down into stories and the individual stories are estimated. Those estimates are summed across teams to arrive at the Feature level estimate.

Feature estimates can be rolled up into Epic estimates in the portfolio backlog. Each of these techniques can be used to estimate job size, the denominator in the Weighted Shortest Job First (WSJF) equation. Keep in mind though, that as story estimates are rolled up to Feature and Epic estimates, the level of precision decreases. Spikes can always be used to provide additional analysis in order to further understand the solution and come up with more accurate estimates.

Relative Estimation

Relative estimation is one of the several distinct flavors of estimation used in Agile teams. It consists of estimating tasks or User Stories, not separately and in absolute units of time, but by comparison or by grouping items that are equivalent. When adopting Agile as a new technique for a team, frequently there will be a large backlog of Stories that need to be estimated all at once. One of the biggest advantages of Agile estimation is that Stories are estimated relative to each other, not on the basis of hourly or daily effort. It's usually clear to a team, regardless of their level of experience, if one story is going to be more difficult than another, even when nobody has any idea how long it may take to complete individual Stories. But going through the process of individual point estimation for a huge list of Stories can be daunting. Relative mass valuation is a quick way to go through a large backlog of Stories and estimate them all as they relate to each other.

To use this approach:

- Write up a card for each Story.
- Set up a large table so the Stories can be moved around easily relative to each other.
- Pick any Story to start, and then get the team to estimate whether they think it is small, medium, or large.
- If it's a large Story, place it at one end of the table. If it's a small Story, it goes at the other end of the table. A medium Story goes in the middle. Now select the next Story and ask the team to estimate if it's more or less effort than the one that you just put down. Position the Story card on the table relative to the previous card, and go to the next card.
- It's possible to go through 100 or more backlog Stories and estimate their relative effort in as little as an hour.

- The next step is to assign points based on the position of the Stories on the table. Start with the easiest Story that is worth assigning points to, and call it a 1.
- Then move up the list of cards, assigning a value of 1 to every Story until you get to one that seems at least twice as difficult as the first one. That Story gets a 2.
- You may need to remind the team not to get caught up in the fine details. The idea is to get a rough point estimate, not a precise order.
- Ultimately, any Story may be completed in any order based on the business value and priority assigned by the Product Owner, so all the team needs to estimate is how many points one Story will take relative to another.

T-Shirt Sizes

Using numbers is the most common approach for estimating points, but sometimes teams find themselves overanalyzing when trying to arrive at a number of points. If you notice that team members are getting caught up in the idea that the number of points associated with a story has anything to do with the number of hours involved in delivering the value of that story, it may be more effective to switch to a non-numerical system like T-shirt sizing.

- With T-shirt sizing, the team is asked to estimate whether they think a Story is extra-small, small, medium, large or extra-large. By removing the implied precision of a numerical score, the team is free to think in a more abstract way about the effort involved in a Story.
- There are some practical issues to consider when adopting T-shirt sizing for Story estimation.
- For one, non-numerical scales are generally less granular. While that can speed up the voting process by reducing the number of options, it may also reduce the accuracy of velocity estimates.
- In addition, the ability to compare Stories with each other can be a little bit more complicated, since there is no clear mathematical relationship between a medium and an extra-small.
- T-shirt size scales also require extra effort on the part of the person coordinating the Agile process. The T-shirt sizes need to be converted to numerical values for the sake of tracking effort over time and charting an estimated velocity for the team.
- For that reason, while T-shirt sizes can be very effective for teams just starting out with Agile, eventually it’s a good idea to move the team toward a more rational numerical scale.

Planning Poker

Planning poker is a game that team members can play during planning meetings to make sure that everybody participates and that every voice is heard.

- To begin, each team member is given a set of cards with numbers on them. The numbers are usually ordered from 0 to 21 using the Fibonacci sequence: 0, 1, 2, 3, 5, 8, 13, and 21.
- Then each story is read aloud. After each story is presented, everybody on the team is asked to hold up the card showing the level of effort that they believe this story represents for the team.

- Initially the estimates may be all over the map, but after a while the team will get a sense of how much effort they all estimate is associated with a typical type of Story.
- Once all the votes are in, the team members with the lowest and highest estimates explain why they chose their scores.
- Frequently, experts with detailed knowledge may be able to tell the rest of the team why a certain story is actually much easier than they thought, or why it may be more difficult than it first appears because of unexpected requirements.
- Through this process, everybody on the team learns more about what's involved in estimating Stories both inside and outside of their specialties, increasing knowledge sharing across the entire team.
- With planning poker, the numbers are significant. A story estimated as a 2 should be about one fourth as difficult as a story estimated as an 8.
- Stories estimated at 20 or higher may be so large that they need to be broken down into smaller Stories before they can be attempted.
- Stories estimated at 1 represent the smallest unit of work.

Appendix A4: Prioritization Techniques

The following sections describe prioritization techniques in greater detail.

Weighted Shortest Job First (WSJF)

One of the most critical and difficult aspects of keeping an Agile development effort operating as efficiently as possible, is the coordination and scheduling, and hence sequencing, of work to be completed in releases and sprints. To assist in this effort, items in the backlog are prioritized so that IAE is getting the most value out of the work being accomplished. This relies on the expectation that items in the backlog are accurately prioritized. IAE uses the WSJF technique for prioritizing Epics and Features in the backlog.

The WSJF is calculated as the *Cost of Delay (CoD) divided by job duration*. Jobs that can deliver the most value (CoD) and are of the shortest duration are selected first for implementation. The following factors contribute to CoD:

User-Business Value: Do our users prefer this over that? What is the revenue impact on our business? Is there a potential penalty or other negative impact if we delay?

Time Criticality: How does the user/business value decay over time? Is there a fixed deadline? Will they wait for us or move to another solution? What is the current effect on customer satisfaction?

Cost of Delay = User Business Value + Time Criticality

WSJF = Cost of Delay / Job Size

It's not essential that these estimates are absolute. We just need to be able to compare backlog items relative to each other; therefore, we can use Fibonacci numbers to estimate the relative cost of delay just as we did for story point estimation. Job duration can be calculated by using the readily available job size, which is calculated for Features (as Feature Size) and Stories (as Story Points) and then stored in the backlog in JIRA. Job size can be estimated for Epics using the Fibonacci method.

Once these values are calculated, a spreadsheet can be used to determine the item with the highest WSJF.

Table 11: Sample Spreadsheet for Calculating Weighted Shortest Job First

Backlog Item	User / Business Value	Time Criticality	Job Size	WSJF
Integrate Data Lake with Landing Page	1	1	3	0.67
User Roles Authorization for Marketplace	3	5	5	1.6

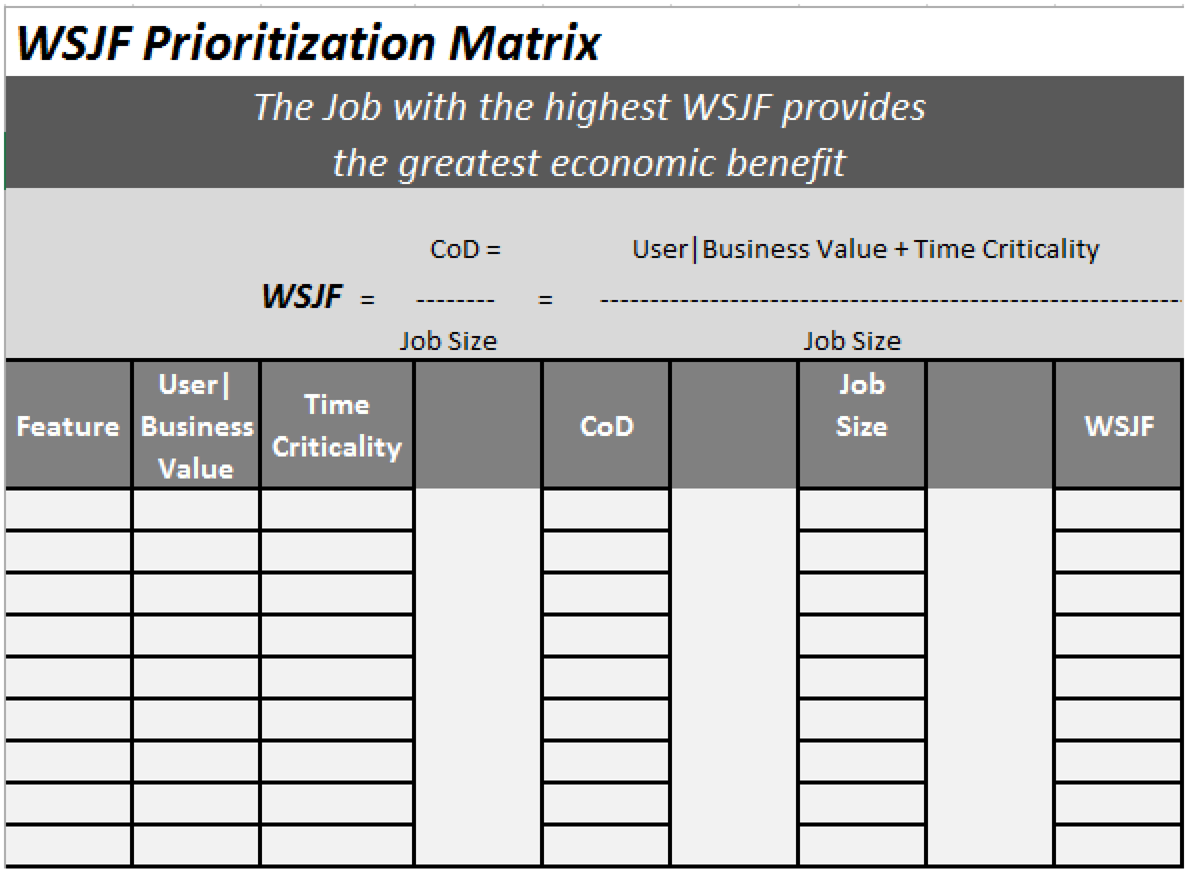
To calculate WSJF, the team rates each backlog item relative to one another for each CoD variable. With relative estimating, do one column at a time, set the smallest item to a "one," and then set the others relative to that item. Higher numbers represent higher value and criticality. When all columns are estimated, divide the CoD by the Job Size. Job Size is a relative estimate of the duration of each item in the backlog. Finally, the item with the highest WSJF also has the highest priority.

Other Prioritization Techniques

Below are other Agile prioritization techniques:

- Monopoly Money
  - Monopoly Money is a technique that involves giving the customer "monopoly money" or "false money" equal to the amount of the project budget and asking them to distribute it among the User Stories under consideration. In this way, the customer prioritizes based on what they are willing to pay for each User Story.
- 100-Point Method
  - The 100-Point Method was developed by Dean Leffingwell and Don Widrig (2003). It involves giving the customer 100 points they can use to vote for the features that they feel are most important.
- Kano Model
  - Kano Model was developed by Noriaki Kano (1984) and involves classifying features or requirements into four categories based on customer preferences: Exciters/Delighters, Satisfiers, Dissatisfiers, and Indifferent.
- Requirement Prioritization Model
  - Requirements prioritization is an essential mechanism of Agile development approaches to maximize the value for the clients and to accommodate changing requirements.
- Relative Prioritization
  - You use planning, ranking and priority fields to specify which work the team should complete first. If you rank user stories, tasks, bugs and issues, all team members gain an understanding of the relative importance of the work that they must accomplish. Ranking and priority fields are used to build several reports.
  - You rank and prioritize work items when you review the backlog for a product or iteration. For more information, see Rank the Product Backlog and Iteration Backlog.

Appendix A5: WSJF Template



WSJF Prioritization Matrix

The Job with the highest WSJF provides the greatest economic benefit

$$\text{WSJF} = \frac{\text{CoD} =}{\text{Job Size}} = \frac{\text{User | Business Value + Time Criticality}}{\text{Job Size}}$$

Feature	User | Business Value	Time Criticality		CoD		Job Size		WSJF

Figure 11: WSJF Prioritization Matrix

Appendix A6: Agile Ceremonies

Below is a list of ceremonies used in the Agile process along with their purposes, key participants, inputs, outputs/deliverables, frequency with which they are held, and the tools used during the ceremony.

Table 12: Agile Ceremonies

Ceremony	Purpose	Key Participants	Inputs	Outputs / Deliverables	Frequency	Tools
PORTFOLIO LEVEL						
GOVERNING BODY MEETING	• Provide overall governance • Announce new business initiatives • Define/Refine strategic themes • Provide strategic direction for IAE process and product development • Review Portfolio Backlog implementation status	• Executive Steering Committee members • Portfolio Management Team • Epic Owners	• Problems with existing solutions • Epics awaiting approval	• Strategic Priorities • New initiatives and opportunities • Proposed Epics • Go/No Go Decisions on new Epics	Quarterly	JIRA
PORTFOLIO MANAGEMENT MEETING	• Identify Epic Owners • Make Go/No Go decision on program driven initiatives/propose d Epics • Allocate resources to implement high priority Epics • Prioritize approved Epics using WSJF • Identify and approve High-Level Portfolio milestones • Guide program execution and governance • Review portfolio, Release, and Sprint implementation status and metrics • Resolve	• Epic Owners • Enterprise Architect • IAE PMO • Portfolio Management Team	• Portfolio Backlog • Portfolio, Program, and Team level status, metrics, and impediments	• Refined and Prioritized Portfolio Backlog • Portfolio Vision • Corrective action and resolutions to impediments • Program Epics	Monthly or As-needed	JIRA

	impediments					
PROGRAM LEVEL						
PRODUCT MANAGEMENT MEETING	• Shepherd proposed Epics until ready for approval • Review and analyze Epic value statements and lightweight business cases • Discuss solutions and alternatives • Establish viability, measurable benefit, development and deployment impact, potential availability of resources • Establish preliminary estimates of opportunity, establish effort and cost of delays • Drive collaboration amongst key stakeholders • Provide quantitative data as basis for decision making • Provide recommendations to the Portfolio Management Team on new Epics • Define and prioritize the Program Backlog • Define and communicate the Program Vision	• Enterprise Architect • Product Management Team • RTE • System Teams • IAE PMO representatives	• New initiatives and opportunities • Proposed Epics • Program Epics	• Lightweight Business Case • Epic Value Statement • Epic Success Criteria • Prioritized Program Backlog Features and priorities • Program Vision • Release Roadmap	Bi-weekly or as-needed	JIRA

	and Roadmap • Work with the Portfolio Management Team to communicate release objectives and Epics • Establish benefits and acceptance criteria for product features • Drive Release content via prioritized features					
PRE-RELEASE PLANNING MEETING	• Review the Program Backlog • Add, revise or remove Program Epics and/or Features • Prioritize Program backlog using WSJF • Review and approve proposed Release Roadmap	• CCB: 24 CFO Agents • Program Management • RTE	• Release Roadmap	• Release Roadmap with a list of approved Features for the next release	Quarterly or as needed	JIRA
RELEASE PLANNING	• Ensure cadence & synchronization to assure value delivery • Establish committed objectives for the next Release (by finalizing Features for the release) • Ensure a realistic implementation plan (estimate Features, identify and resolve program-wide dependencies and impediments) • Establish transparency and collaboration	• RTE • Product Management • Release management • PMO • DevOps/IV & V • Vendors	• Program Vision • Release Roadmap • Features backlog approved by CCB	• Release objectives • Prioritized Team Backlog • Features sequenced in individual sprints • Final Release Roadmap and Schedule	Quarterly	JIRA

	across the teams and organization (agree on high-level release schedule, and secure team's commitment to the release, release confidence vote)					
SCRUM OF SCRUMS	• Collaboration among Scrum Masters to discuss their work, focusing especially on areas of overlap, dependencies and integration	• RTE • Scrum Masters (Vendors)	• Team update, dependency/ impediments	• Planned accomplishments • Resolution to impediments	Once a week or as needed	JIRA
PRODUCT SCRUM OF SCRUMS	• Collaboration among Product Owners to discuss their work, focusing especially on areas of overlap, dependencies and integration	• RTE • Product Owners	• Team update, dependency/ impediments	• Planned accomplishments • Resolution to impediments	Twice a week (Wed, Friday), or as needed	JIRA
RELEASE DEMO	• Provide the program-level view of all new Features delivered by Agile Teams • Held at the end of each release, and optionally at the end of each sprint when new Features are completed	• Product Management • System Teams • Enterprise Architect • Epic Owners • RTE • Scrum Masters • Agile Teams • DevOps & IV & V	• Working software with developed Features	• Customer feedback on the demo	Quarterly	System with Working Software
RELEASE RETROSPECTIVE	• Engage in the Inspect and Adapt process to: ○ Evaluate performance of current release ○ Conduct retrospective	• Product Management • System Teams • Enterprise Architect • Business Owners • RTE • Scrum Masters	• Release Metrics	• Corrective Action Plan • Set of improvement user stories	Quarterly	JIRA

	and problem-solving workshop ○ Reflect how future releases can be more effective	• Agile Teams • DevOps & IV & V				
TEAM LEVEL						
TEAM BACKLOG GROOMING	• Review Team Backlog & Implementation status • Ensure that Acceptance Criteria exist for each story • Verify Story estimates • Plan to build/maintain Development Infrastructure • Plan to build/maintain Architectural Runway • Prepare for Sprint Planning	• Product Owner • Scrum Master • Agile Team	• Team backlog	• Refined Team Backlog	Weekly or as needed	JIRA
SPRINT PLANNING	• Time-boxed meeting to commit user stories for the Sprint • Discuss stories and estimate story points	• Product Owner • Scrum Master • Agile Team	• Team Backlog • Release objectives • Team Velocity	• Sprint Goals • Committed Sprint Backlog • Task list	Bi-Weekly	JIRA
DAILY SCRUMS	• Team shares information about previous day's progress • Coordinate current day's activities • Identify dependencies and report impediments	• Product Owner • Scrum Master • Agile Team	• Individual activities, dependency/ impediments	• Activities coordinated across the team • Impediments identified	Daily	N/A

SPRINT DEMO	• Demonstrate progress to Program Management and other stakeholders • Demonstrate completed user stories • Review sprint metrics	• Scrum Master • Product Owner • Agile Team • Other stakeholders	• Working Software • Sprint metrics • Completed sprint backlog	• Product Owner feedback • Accepted and Rejected stories • Enhancement list	Bi-Weekly	The System with Working Software
SPRINT RETROSPECTIVE	• Team reflects on how to become more effective • Discuss what went well, what didn't, and what the team can do better next time	• Scrum Master • Agile Team • Product Owner	• Sprint metrics • PO, stakeholder feedback • Action items status from previous sprint	• Lessons learned report • What/how to do better next time • Root cause analysis report • Action items for upcoming Sprint • Owner of each action item	Bi-Weekly	JIRA

APPENDIX A7: AGILE TOOLS

This section describes the two key Atlassian tools (JIRA and Confluence) that are used to help manage the Agile process. ***Attachment GSA IAE CSP Architecture*** (listed in the front matter of this document) contains a list of the tools and software provided by the IAE Common Services Platform.

JIRA

JIRA is the Agile Lifecycle management tool for IAE currently being used as a Backlog Management tool. JIRA does much more than just track backlog items, though: It allows Agile Teams to manage IAE development via Sprints and releases from the task level all the way up to the Portfolio Epic level.

Please refer to the IAE JIRA Standard Operating Procedures (SOPs) document for additional information on how to use all of the JIRA tools to manage the Portfolio, Program and Team backlogs.

CONFLUENCE

Atlassian's Confluence is a collaboration tool that provides IAE with a centralized repository allowing users to organize, create, share, discuss and search project data and documents. Confluence is a wiki, software that runs on a server and publishes Web pages that you can read via a Web browser.

Some of the collaboration capabilities that Confluence provides include:

- Share meeting notes
- Share files
- Make collaborative decisions
- Share links
- Assign tasks
- Share calendars

Confluence is also an Agile development tool that allows users to:

- Store requirements
- Create JIRA issues
- Link to JIRA
- Publish reports
- Track Releases
- Create Retrospective templates
- Store lessons learned

Finally, Confluence is a knowledge base with a simple setup. It allows users to:

- Create knowledge articles
- Share knowledge and best practices internally within teams or organization

APPENDIX A8: AGILE ENGINEERING PRACTICES

The following Agile Engineering Practices support the development of high quality software. These practices are adopted from the SAFe Code Quality practices.

Continuous integration

Continuous Integration is the software development practice that requires team members to integrate their work frequently.

- Every developer integrates at least daily, which leads to multiple integrations each day.
- Integrations are verified by an automated build that runs automated regression tests to detect integration errors as quickly as possible.
- This approach leads to significantly fewer integration problems and enables development of high quality software more rapidly.

TEST FIRST

"Test First" adheres to both Acceptance Test-Driven Development (ATDD) and Test-Driven Development (TDD) practices.

- With ATDD, the acceptance tests are built before developers begin coding.
- With TDD, developers build the test first and then develop functionality until the code passes the test.
- This is typically followed by refactoring the code to improve maintainability.
- As the software grows, new tests are added to the suite of tests that are run during the Continuous Integration process to ensure a functioning system is continuously maintained.
- This practice requires testing to be automated.
- The types of testing executed during Continuous Integration include but are not limited to the following:
  - Unit tests
  - Functional tests
  - Continuous inspection of code quality
  - Security vulnerability tests
- Characteristics of good unit tests include fast execution, isolated (not dependent on other tests), and ability to leave the system under test unaltered.
- Continuous Integration also applies to system level testing and is executed at least once every sprint.
- System testing includes end-to-end system testing and performance testing.
- All test code, scripts and data are required to be placed under version control in the code repository.

REFACTORING

Refactoring is the practice of clarifying, simplifying and improving the existing design of the code as part of the Agile software development process.

- Refactoring prevents the system from becoming unmaintainable as the software grows.
- This practice keeps the code easy to maintain and extend.
- Refactoring is only possible if the automated tests are developed following the Test First method.
- A comprehensive set of regression tests is run after each step during refactoring to provide immediate feedback to the developer.

Pair Work

Pair Work is inspired by eXtreme Programming's (XP's) pair programming practice where two programmers work together in front of a workstation to enable continuous collaboration, knowledge sharing, and peer review of the code.

- Pair Work expands on this practice in that team members are encouraged to work together in pairs whenever it makes sense.
- Pair Work could be between developers, testers, business analysts, and other team members when collaboration and knowledge sharing would help deliver value faster.

Collective Ownership

Collective Ownership encourages everyone to contribute to the project.

- Any developer can feel free to change or refactor any line of code to add functionality or improve the design.
- This practice avoids the bottleneck situation when only one developer has in-depth understanding of a section of the code.
- This convention is made possible by the comprehensive set of unit tests that gives the developers the confidence to touch the code written by others.

DevOps

The IAE lifecycle will be implemented based on the IAE DevOps standards (TBD).

- DevOps is a new term that promotes collaboration between development and operations staff throughout all stages of the development lifecycle through the deployment and delivery of value to the end user.
- The IAE DevOps team will establish and automate the entire environment creation and deployment process.
- Deployment readiness is continuously maintained to enable release of value based on customer demand.

APPENDIX A9: AGILE ARCHITECTURE PRACTICES

Architectural Epics flow through the same IAE Agile Framework processes as Business Epics. The only difference is that Business Program Managers manage business Epics and Architecture Program Managers manage architecture Epics.

An ***Architectural Runway*** provides a roadmap for implementing architectural features. The key features are:

- Architectural teams iterate like every other Agile team on the program.
- Credit goes to working code, not models and designs.
- Time is of the essence. It should take no more than a few iterations to prove the new architecture.
- Architecture code needs to stay ahead of business features dependent on new architecture.

Appendix A10: Agile Architecture Principles

IAE adheres to SAFe Architectural Principles that promote "every team deserves to see the bigger picture" and "every team is empowered to design their part."

Principle 1: Design Emerges. Architecture is Collaboration

While an Agile Manifesto principle states "the best architectures, requirements, and designs emerge from self-organizing teams," intentional architecture has a place in large, complex systems to provide guidance and technical governance to Agile Teams, while still empowering the teams to design their own parts. (Refer to AE framework documentation for additional information on architectural framework and standards.)

- **Emergent Design**: The evolutionary process of discovering and extending the design only as necessary to implement and validate the next increment of functionality.
- **Intentional Architecture**: For a large, complex system, it is impossible for teams to anticipate changes that may well occur outside their environment, nor for individual teams to fully understand the entire system. The solution is a set of purposefully built architectural artifacts to enhance the understanding of the requirements to support the solution design and implementation.
- **Collaboration**: The right balance of emergent design and intentional architecture drives effective evolution of the system.

Principle 2: the bigger the system, the longer the runway

The architectural runway exists when enterprise platforms have sufficient technology infrastructure to support the implementation of the highest priority Epics and Features in the backlog without excessive, delay-inducing redesign. In order to achieve some degree of runway, the enterprise must continually invest in extending existing platforms as well as building and deploying new platforms.
Architecture Epics must be decomposed into architecture Features that are implemented as part of individual releases. Architecture runway can be exposed to the consumer when sufficient capabilities exist to support the implementation of near-term business Epics and Features.

Principle 3: Build the Simplest Architecture that can possibly work

Agile promotes using the simplest architecture and design that can support current requirements. The right balance of intentional architecture and emergent design allows for generalization of solution elements to address common problems at the component, system or enterprise level.

Principle 4: When in doubt, code, or model it out

Making good design decisions helps avoid unnecessary refactoring. Use of rapid prototyping and short spikes provide feedback with objective evidence, which can help solidify the design. Technical teams can use a healthy mix of spikes, prototyping and modeling to better understand potential impacts prior to implementation.

Principle 5: They build it, they test it

Testing system architecture involves testing the system's ability to meet its larger scale functional, operational, performance and reliability requirements. To do this, teams must typically build an automated testing infrastructure that enables ongoing system-level testing. As the system evolves, the testing approaches, testing frameworks and test suite must evolve with it.

Principle 6: There is no monopoly on innovation

Architecture is a collaborative effort, which helps foster a culture whereby innovation can come from anyone and anywhere. One of the responsibilities of the enterprise architect is to foster an environment where innovation ideas and technology improvements that emerge at the team level are leveraged in future implementation.

Principle 7: Implementing architectural flow

IAE will continuously improve the process of implementing large architectural initiatives that impact the runways of multiple systems. Cross-cutting Architectural Epics must be managed properly through the Epic Kanban process to support the continuous delivery of business value.

B. Portfolio Information

Appendix B1: Portfolio Key Roles and Responsibilities

The following are the key Roles and Responsibilities at the Portfolio Level.

Table 13: Key Roles and Responsibilities (Portfolio Level)

Role	IAE Representatives	Responsibility
IAE Governing Body	• ACE • PCE • FACE • OMB Directives • OCIO	• Defines Strategic Themes and business objectives that connect the portfolio to the business strategy • Elaborates on competitive differentiation from the current state to a future state • Participates in business case analysis, cost estimation, prioritization and Go/No Go decision meetings
Portfolio Management Team	• IAE Assistant Commissioner • IAE Deputy Assistant Commissioner • IAE Directors • RTE	• Program Portfolio Management has the highest fiduciary decision-making responsibility • Executives with market knowledge, technology awareness, and understanding of financial constraints and market conditions analyze, justifies business case, initiatives • Drives product and solution strategy; manage investment • Participates in business case analysis, cost estimation, prioritization and Go/No Go decision meetings
Epic Owner*	• Program Manager (Business/Technical) • Enterprise Architect	• Owns Business or Architectural epics • Defines, analyzes, and move selected epics into implementation • Works with release trains to realize business benefits of the epic • Participates in business case analysis, cost estimation, prioritization and Go/No Go decision meetings
Enterprise Architect	• IAE IT Director	• Works with business stakeholders and system architects to drive holistic implementation across enterprise • Drives key initiatives and strategy for maintaining enterprise architectural runway • Participates in business case analysis, cost estimation, prioritization and Go/No Go decision meetings

* IAE business and technical Program Managers wear multiple hats, performing the roles of Epic Owner, Product Manager and Product Owner.

APPENDIX B2: PORTFOLIO LEVEL PROCESS DETAILS

The following diagram represents the Portfolio Level Process, which is at the highest level in the framework.

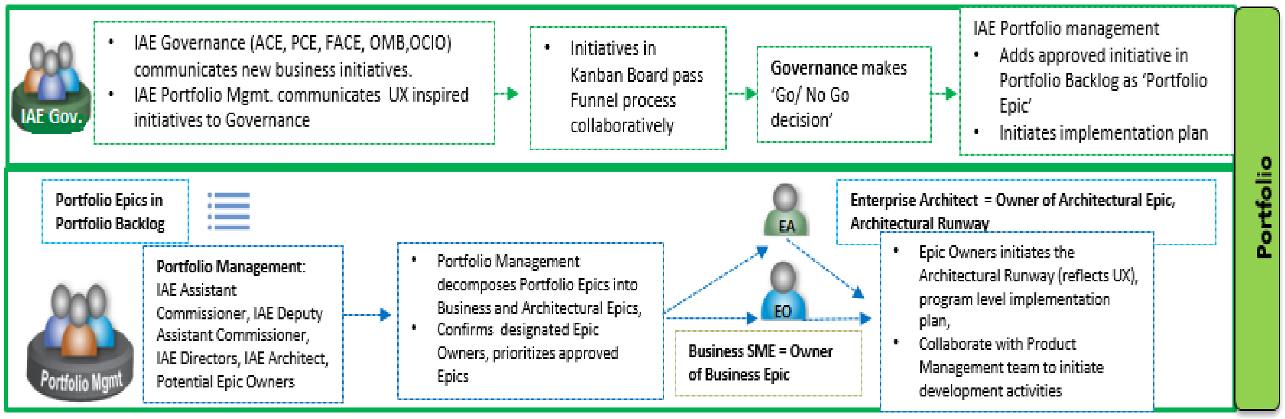

IAE Gov.
IAE Governance (ACE, PCE, FACE, OMB,OCIO) communicates new business initiatives.
IAE Portfolio Mgmt. communicates UX inspired initiatives to Governance
Initiatives in Kanban Board pass Funnel process collaboratively
Governance makes 'Go/ No Go decision'
IAE Portfolio management
Adds approved initiative in Portfolio Backlog as 'Portfolio Epic'
Initiates implementation plan
Portfolio
Portfolio Epics in Portfolio Backlog
Portfolio Mgmt
Portfolio Management: IAE Assistant Commissioner, IAE Deputy Assistant Commissioner, IAE Directors, IAE Architect, Potential Epic Owners
Portfolio Management decomposes Portfolio Epics into Business and Architectural Epics,
Confirms designated Epic Owners, prioritizes approved Epics
EA
EO
Business SME = Owner of Business Epic
Enterprise Architect = Owner of Architectural Epic, Architectural Runway
Epic Owners initiates the Architectural Runway (reflects UX), program level implementation plan,
Collaborate with Product Management team to initiate development activities

FIGURE 12: PORTFOLIO LEVEL PROCESS

At this level all strategic initiatives, business and technology (architecture) are vetted to ensure that they align with the overall enterprise strategy to provide the greatest benefit and value for the organization and its key stakeholders. A comprehensive Portfolio Backlog is created at this level with the input of the stakeholders. The portfolio backlog is prioritized, and Epics are created after an initial analysis. The final list of Epics is prioritized, and budget and resources are allocated to the approved initiatives (Epics). The ART formation is initiated at this level, and Epics are allocated to the ART.

GOVERNING BODY COMMUNICATES NEW INITIATIVES

The Governing Body is responsible for establishing the enterprise vision and strategy direction. The Governing Body defines strategic themes and new business initiatives that influence Epics evaluation, program vision, and the roadmap.

These new initiatives have multiple sources including:

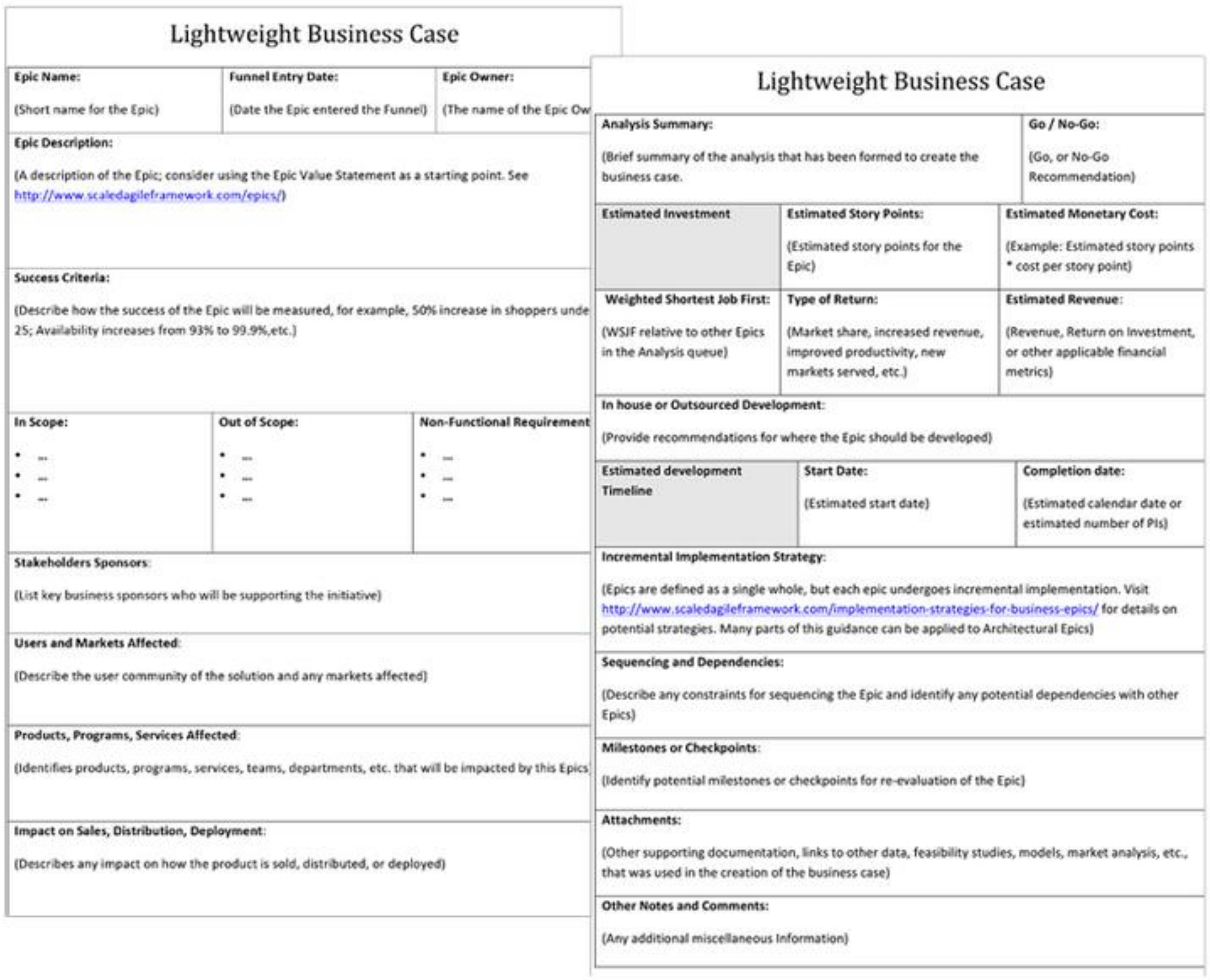
- Enterprise strategic themes
- Portfolio vision
- Changes in the marketplace
- Identified need for substantive cost savings or operational efficiencies
- Problems with existing solutions that are hindering business performance.

PORTFOLIO BACKLOG FORMATION

The Portfolio Backlog provides a holding mechanism for the upcoming Business and Architectural Epics. The creation of all "big business and architectural ideas" are introduced, also known as the

Problem/Solution Needs Identification, from specific, itemized business objectives to the evolving enterprise business strategy.

Epics and Lightweight Business Cases/Charters provide visibility and economic justification for upcoming, cross-cutting work. Business or Architectural Epics are defined and analyzed, each supported by a lightweight business case. Developed by Program Managers, lightweight business cases and charters provide for reasoning, analysis and prioritization while avoiding over-specificity.



Lightweight Business Case

Epic Name: (Short name for the Epic)	**Funnel Entry Date:** (Date the Epic entered the Funnel)	**Epic Owner:** (The name of the Epic Ow
Epic Description: (A description of the Epic; consider using the Epic Value Statement as a starting point. See http://www.scaledagileframework.com/epics/)		
Success Criteria: (Describe how the success of the Epic will be measured, for example, 50% increase in shoppers unde 25; Availability increases from 93% to 99.9%,etc.)		
In Scope: • … • … • …	**Out of Scope:** • … • … • …	**Non-Functional Requirement** • … • … • …
Stakeholders Sponsors: (List key business sponsors who will be supporting the initiative)		
Users and Markets Affected: (Describe the user community of the solution and any markets affected)		
Products, Programs, Services Affected: (Identifies products, programs, services, teams, departments, etc. that will be impacted by this Epics)		
Impact on Sales, Distribution, Deployment: (Describes any impact on how the product is sold, distributed, or deployed)		

Lightweight Business Case

Analysis Summary: (Brief summary of the analysis that has been formed to create the business case.		**Go / No-Go:** (Go, or No-Go Recommendation)
Estimated Investment	**Estimated Story Points:** (Estimated story points for the Epic)	**Estimated Monetary Cost:** (Example: Estimated story points * cost per story point)
Weighted Shortest Job First: (WSJF relative to other Epics in the Analysis queue)	**Type of Return:** (Market share, increased revenue, improved productivity, new markets served, etc.)	**Estimated Revenue:** (Revenue, Return on Investment, or other applicable financial metrics)
In house or Outsourced Development: (Provide recommendations for where the Epic should be developed)		
Estimated development Timeline	**Start Date:** (Estimated start date)	**Completion date:** (Estimated calendar date or estimated number of PIs)
Incremental Implementation Strategy: (Epics are defined as a single whole, but each epic undergoes incremental implementation. Visit http://www.scaledagileframework.com/implementation-strategies-for-business-epics/ for details on potential strategies. Many parts of this guidance can be applied to Architectural Epics)		
Sequencing and Dependencies: (Describe any constraints for sequencing the Epic and identify any potential dependencies with other Epics)		
Milestones or Checkpoints: (Identify potential milestones or checkpoints for re-evaluation of the Epic)		
Attachments: (Other supporting documentation, links to other data, feasibility studies, models, market analysis, etc., that was used in the creation of the business case)		
Other Notes and Comments: (Any additional miscellaneous Information)		

FIGURE 13: EPIC LIGHTWEIGHT BUSINESS CASE

EPIC UNDERSTANDING AND ALTERNATIVES (KANBAN WORKSHOP)

The purposes of the Kanban Workshop are to provide visibility and guide new initiatives through the Kanban process. The Kanban workshop reviews and analyzes new initiatives, develops Epic value statements, and develops lightweight business cases. The Kanban Workshop prepares final recommendations of proposed Epics for final approval by the Portfolio Management Team. Once the Portfolio Management Team approves an Epic, it is added to the Portfolio Backlog.

All initiatives are captured and fed into the Kanban system by the IAE Portfolio Management team. Kanban is a method for visualizing and managing work that contains a series of defined states through which the work moves. There are usually specific Work In Progress (WIP) limits for each state, which change only as necessary to improve flow. The Kanban system at the Portfolio level is a lightweight approach for managing the flow of Epics. A Kanban system is used because it provides the following:

- Transparency to the initiatives being developed
- Structure to the analysis and decision making process
- Assurance that expectations for implementation of initiatives align with the realities of capacity limits
- A mechanism to drive collaboration amongst key stakeholders, and
- A quantitative, transparent basis for decision making.

The Kanban system represents a collaborative effort among IAE Portfolio Management, the Enterprise Architect and the Epic Owner.

The portfolio Kanban system is composed of five states:

- Funnel
  - Initiatives are first captured.
  - All ideas are considered, elaboration is not required, and any mechanism can be used for capturing the idea.
  - No WIP limit at this stage since cost of capture is minimal.
  - On a periodic cadence, set by the Portfolio Management team, these big initiatives or Epics are discussed, and the ones that meet the decision criteria are moved to the Review queue.
- Review
  - Justification of Epics.
  - Epics are roughly sized.
  - A rough estimation of value is established.
  - Epic value statements, which provide additional Epic details, are provided.
  - Sources of business benefit are identified.
  - Review Epics are discussed periodically, and there may even be some very preliminary investigation performed. Because of the increased investment, WIP limits are imposed.
  - Review Epics are assigned a Cost of Delay using the WSJF methodology.
  - Those Epics at the top of the queue are pulled into the Analysis state as soon as capacity is available.
- Analysis
  - Requires further analysis and investment.
  - Interim Epic Owner is identified who will move the Epic through the Kanban process.
  - Active collaboration is initiated between enterprise and system architects, product management and key stakeholders.
  - Analysis of the Epic includes work to determine viability, measurable benefit, development and deployment impact, and potential availability of resources.
  - Exploration of design and implementation alternatives occurs, as well as options for internal development and/or outsourcing.

    - The Epic Owner creates a Lightweight Business Case, which captures the results of the analysis and determines the priority of each Epic.
    - Epics in this queue are WIP-limited due to the fact that required resources (Epic Owner, Enterprise Architect, and solution development) are scarce as well as the fact that these Epics are going to require a substantial upcoming investment.
    - Based on the results of the business case, the Portfolio Management Team makes a Go/No Go decision on whether the Epic should move on to the Portfolio Backlog.
- Portfolio Backlog
    - Epics approved by the Portfolio Management Team.
    - Continuous prioritization of approved Business and Architectural Epics using WSJF.
- Implementing
    - Decompose Business and Architectural Epics into features.
    - Transition to the Program Team.
    - Analyst support on pull (get more Epics from portfolio backlog) basis.
    - WIP limited by capacity.

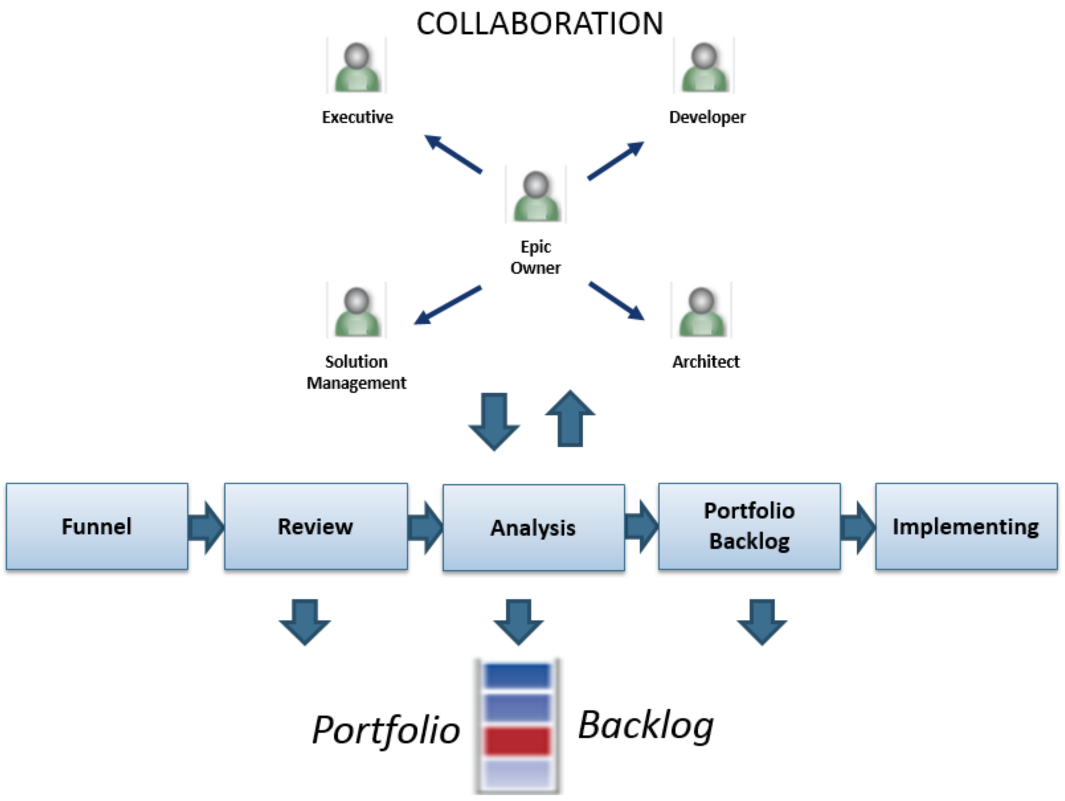


FIGURE 14: KANBAN WORKSHOP WORKFLOW

PRIORITIZATION OF PORTFOLIO BACKLOG (PORTFOLIO BACKLOG GROOMING)

The Portfolio Management Team is responsible for:

- Managing investments
- Driving Epic development
- Monitoring Epic implementation progress
- Grooming and prioritizing Epics in the Portfolio Backlog

The prioritization of Epics ensures that the highest priority Epics are promoted to implementation when there is sufficient capacity from the ARTs. The Portfolio Management Team provides direction and guidance on the overall solution strategy for the implementation of Portfolio Epics and the execution of releases.

As part of the portfolio Backlog Grooming process, the Portfolio and Product Management teams, Epic Owners and Enterprise Architect review the backlog and prioritize the Epics using the WSJF technique. The highest priority Epics are pulled from the Portfolio backlog and moved to the Program Backlog for implementation when a Release Train has available capacity.

- Portfolio backlog
    - Approved Epics are added to the Portfolio Backlog as Portfolio Epics by the IAE Portfolio Management team.
    - The backlog is reviewed on a periodic cadence, and when capacity becomes available on an ART, an Epic can move to implementation.
    - Prior to being scheduled for implementation, each Epic goes through additional reasoning.
    - The Portfolio Management Team, along with the Epic Owner and Enterprise Architect, analyzes the Portfolio Epics and decomposes them into Business and Architecture Epics.
    - Epic Owners' assignments are finalized based on the type of Epic.
    - Business SMEs own business Epics, while the Enterprise Architect owns architectural Epics.
    - Perform a final verification of Epic priority using WSJF.
    - Highest priority items get pulled from the Portfolio Backlog when there is sufficient program capacity on one or more ARTs.

PORTFOLIO MONITORING (HEALTH CHECK)

The Portfolio Management Team reviews overall status, dashboards and metrics on the implementation Epics, Features and the execution of Releases and Sprints. The Portfolio Management Team takes action to help resolve any impediments escalated from Releases and Teams. Corrective actions are taken to ensure continuous delivery of value.

The retrospective will address the following:

- What is the status of the current release?
- What impediments must we address to facilitate progress?
- Are we likely to meet the release objectives, and if not, what adjustments are required?

APPENDIX B3: IAE PROGRAM MANAGER PORTFOLIO CHECKLIST

TABLE 14: IAE PROGRAM MANAGER PORTFOLIO CHECKLIST

IAE Program Manager Portfolio-Level Checklist	Status
Prior to Approval	
Work with stakeholders and subject matter experts to define the epic, its potential benefits, and establish the cost of delay. Identify business sponsors.	
Work with the development teams to size the epic and provide input for economic prioritization based on WSJF.	
Define epic success criteria	
Shepherd the epics through the Epic Kanban system and create the lightweight business case.	
Prepare to present the business case to the Portfolio Management Team for a Go/No Go decision.	
Presenting the Epic	
The epic owner has the primary responsibility for presenting the merits of the epic to Portfolio Management Team and the Governing Body.	
After Approval - Implementation	
Work with Product Management to split the epic into Program Epics and features and prioritize them in the Program Backlogs.	
Provide guidance to the release train on the epic context of the target features.	
Participate in Release Planning, Release Demo, and Release Retrospective whenever there is critical activity related to the epic.	
Work with Agile teams that perform research spikes, create proof of concepts, mockups, etc.	
Coordinate and synchronize epic-related activities with functions in sales, marketing, and other business units.	
Understand and report on progress of the epic with key stakeholders.	

C. Program Information

Appendix C1: Program Level Roles and Responsibilities

The following are the key Roles and Responsibilities at the Program Level.

Table 15: Key Roles and Responsibilities (Program Level)

Role	IAE Representatives	Responsibility
CCB	• 24 CFO Agents	• Reviews product feature list/ proposed Release Roadmap • Adds/Removes product feature • Provides Go/No Go Decision
Product Management	• IAE Directors • Program/Product Owners (Business/Technical)	• Serves as content authority for the train • Continuously interacts with customers and stakeholders to define and prioritize the Program Backlog • Owns the Vision and communicates the Roadmap • Defines product features and acceptance criteria • Works to optimize feature delivery to balance the enterprise's technical and economic objectives
Release Management	• RTE • OSM • IT Operations/Security • DevOps • Vendor PM • Product Management • PMO	• Ensures the organization's release governance are defined and understood • Owns release planning • Facilitates and negotiate release content • Ensures Inspect & Adapt, improve program productivity, quality and release process • Communicates release status
RTE	• RTE by GSA IT	• Facilitates release planning readiness and the Release Planning meeting • Assists with program execution and tracking of metrics • Publishes release objectives for visibility and transparency • Facilitates Scrum of Scrums and Release Retrospective meetings • Escalates impediments and helps manage dependencies • Ensures collaboration within and across trains

Role	IAE Representatives	Responsibility
		• Promotes program-level code quality best practices • Drives program-level continuous improvement • Facilitates deployment of working products • Manages Program Risk and Mitigation Strategies
System Team	• Development Representatives • Systems Engineering • IT Security Team • Deployment/DevOps team	• Builds the development infrastructure and manages environments • Supports full system integration • Performs end-to-end system and performance testing • Stages and supports the Release Demo • Determines and establishes the program branching strategy • Assists with build, test and deployment automation strategies and adoption • Supports system-level continuous integration
UX/UI Team	• User Experience Product Owners	• Provides guidance for a consistent user experience • Provides Agile Teams with increments of UI design and UX guidelines to ensure a consistent user experience • Validates user experience via user testing
Outreach and Stakeholder Management	• OSM members	• Coordinates with the Product Management team to incorporate requirements information into the backlog, derived from Focus Groups • Owns the IAE branding • Participates in Release Planning, Release Demo and Release Retrospective ceremonies • Participates in Sprint Review
DevOps	• IT Operations • DevOps team	• Maintains all environments (Dev/Test/Pre-Prod/Prod) in product lifecycle to product quality working code • Automates deployment process and perform deployment to all environments including production • Establishes an effective deployment process • Applies continuous improvement, continuous integration and continues delivery to the deployment process

APPENDIX C2: PROGRAM LEVEL PROCESS DETAILS

At the Program level, the ARTs take over responsibility for implementing Epics, Features and User Stories. All of the ARTs running within the enterprise work in a synchronized cadence. Coordinating efforts among multiple teams in regular, repeatable, short intervals leads to decreased variability in the work and results in a more efficient, dependable, reliable and adaptable work product.

The following diagram represents the Program Level Process, which is at the midlevel in the framework.

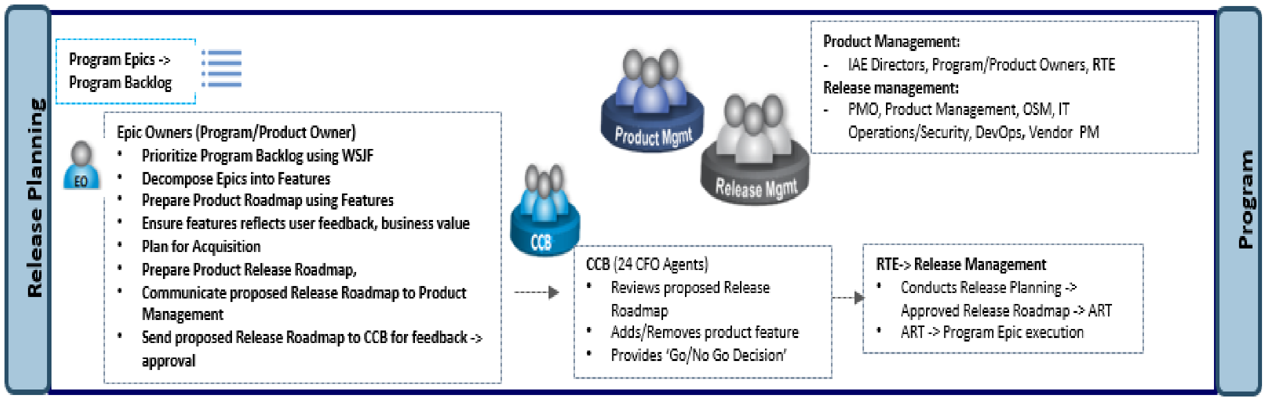

Release Planning
Program Epics -> Program Backlog
EO
Epic Owners (Program/Product Owner)
• Prioritize Program Backlog using WSJF
• Decompose Epics into Features
• Prepare Product Roadmap using Features
• Ensure features reflects user feedback, business value
• Plan for Acquisition
• Prepare Product Release Roadmap,
• Communicate proposed Release Roadmap to Product Management
• Send proposed Release Roadmap to CCB for feedback -> approval
CCB
Product Mgmt
Release Mgmt
Product Management:
- IAE Directors, Program/Product Owners, RTE
Release management:
- PMO, Product Management, OSM, IT Operations/Security, DevOps, Vendor PM
CCB (24 CFO Agents)
• Reviews proposed Release Roadmap
• Adds/Removes product feature
• Provides 'Go/No Go Decision'
RTE-> Release Management
• Conducts Release Planning -> Approved Release Roadmap -> ART
• ART -> Program Epic execution
Program

FIGURE 15: PROGRAM LEVEL PROCESS OVERVIEW

PROGRAM BACKLOG GROOMING

The Program backlog is aligned with the Program level of the enterprise and is the single, definitive repository for all upcoming work anticipated to advance the ART. The program backlog provides value to the enterprise by aligning the development team and stakeholders to a common mission via a single backlog. IAE maintains the program backlog using JIRA.

The sources of the Program backlog include the Portfolio Backlog's Business and Architectural Epics, feedback from customers/stakeholders, Architectural Features and inputs from Agile Teams. Epics can also originate from other sources at the Program level. Epic Owners decompose program Epics into Features (and possibly sub-Features) and prioritize them using WSJF.

The Product Management team is responsible for developing, prioritizing and maintaining the program backlog, with the Program Manager being designated the owner of the backlog. System architects provide architectural input into the program backlog via architectural Features. They work with Product Management to ensure there is enough runway to meet the needs of the upcoming business Features. The Product Management team members refine the Program Backlog.

Epics at the Program level are decomposed into Features and sub-Features in the Program Backlog during the Backlog Grooming process, primarily by the Product Management Team, Epic Owners and the System Architect, but also with assistance from the System Team, Release Management, and DevOps/IV&V.

The Backlog Grooming is used to prepare for the Release Planning ceremony. During Backlog Grooming, existing Features are reviewed and elaborated upon, including defining acceptance criteria and establishing estimates. Larger Features are evaluated for breaking down into smaller Features or sub-Features. Features are also evaluated to determine if they need to be decomposed into related architectural Features. In addition, the capacity that can be allocated for these architectural Features in upcoming releases is determined. Finally, the Features and Sub-Features in the Program Backlog are prioritized using the WSJF approach.

Change Control Board (CCB) Approval

The CCB, composed of 24 CFO Agents, serves as the content authority that is empowered to decide what gets implemented on the ART. It reviews, edits and approves Features to be implemented in the upcoming release via the approval of the Release Roadmap.

Once the Features and sub-Features are defined and prioritized in the Program Backlog, the Product Management team, in consultation with Agile Teams and the Epic Owner, drafts a release roadmap and presents it to the CCB for feedback and approval. The Product Management team also communicates the proposed release roadmap to Release Management, the Outreach and Stakeholder Management (OSM) Team, and DevOps, and collects feedback. The CCB vets the Feature list, modifying it as necessary until approval can be obtained.

The Change Control Board has several release types, which are listed in the table below.

Table 16: Release Types

Release	Content	Schedule	Approver
Major	CCB-Approved Features that are developed in the Release Increment.	• End of Release • Completed features is ready to be deployed into the production environments	CCB
Minor	• Incremental Feature Delivery • Release valve for Emergency or Urgent Change Requests • Infrastructure components	Scheduled on as needed basis	CCB/Product Management

Release Planning

Agile releases run on a three-month cadence of Release Planning -> Implementation -> Release Demo -> Release Retrospective. Release Planning is the ceremony that ensures the program stays on this rhythm of continuous development and delivery. It takes place at the beginning of each release. This ceremony is facilitated by the RTE and supported by all members of the program. This face-to-face ceremony level sets everyone's understanding of the program's vision and roadmap and secures the development teams' commitment to the objectives for the upcoming release. The development teams will assess and mitigate technical dependencies across teams. This high level of collaboration ensures the resulting plan is realistic and implementable.

Once CCB approval of the release roadmap is obtained, the release planning ceremony can be conducted. The release planning ceremony is the key event in the release process where the entire team, stakeholders and business owners meet face-to-face in order to come to an agreement on the objectives of the upcoming release. The release planning ceremony reinforces a cadence-based synchronization among ARTs, promotes development of a realistic implementation plan, and establishes transparency and collaboration across the teams and organization. The meeting is facilitated by the RTE and uses a standardized agenda that includes a presentation of vision, team planning breakouts, and commitment to release objectives for the next release cycle. Each team does the following:

- Creates its plan based on the CCB-approved Features list and release roadmap
- Estimates its capacity (velocity for each sprint); develops and refines stories needed to realize each Feature
- Identifies risks and dependencies
- Develops a set of team objectives, which are aligned to the release objectives.

Identified program risks and impediments are addressed between the entire group and management. A confidence vote is taken. If agreement cannot be reached, the plan is reworked until participants reach consensus. The key outputs of the process are the team and release objectives, a release plan with Features sequenced in individual sprints and related dependencies, and the vote of confidence from the participants. Following the release planning session, the roadmap is updated, and program Features are moved to the individual team backlogs to become part of the ARTs.

Scrum of Scrums (Program Risk Management)

The Scrum of Scrums is used to monitor the progress of the teams during development of the release and to manage risk. The RTE, also known as the Chief Scrum Master, facilitates the scrum of scrums meeting with all the teams' Scrum Masters. The Scrum of Scrums ensures teams are meeting their sprint objectives, removing any problems that are causing delays or confusion, and raising the level of awareness of dependencies between the Agile teams.

- Scrum of Scrums (15 – 30 Minutes)
- A scrum at the program level to gain insights into team progress and program impediments
- Insights gained and program impediments requiring escalation are brought to the Release Management Meeting
- Who should attend: RTE, Scrum Masters, Subject Matter Experts as needed, Program Managers
- Frequency: Weekly or more frequently, as conditions require
    - Recommended to meet two times a week
- Attendance is mandatory. If someone can't make it, a proxy must attend and report
- Distributed teams – Attend a time when all can call in or attend in person
- The RTE has primary responsibility for communicating any major blocks, challenges, or impediments to key stakeholders and the Release Management Team
- Checklist:
    - What did your team accomplish since the last meeting?
    - What will your team accomplish between now and the next meeting?

- Are there any blocking issues?
- Are you about to put a block someone else's way? (Meet after/Problem solving)
    - Affected parties stay for problem solving
    - Timebox: As long as it takes

Once a risk is identified, you need to decide what to do with it. The IAE Agile Framework uses ROAM (Resolved, Owned, Accepted, and Mitigated).

ROAM Model

- **Resolved** – The risk has been answered and avoided or eliminated.
- **Owned** – The risk has been allocated to someone who has responsibility for doing something about it.
- **Accepted** – The risk has been accepted and it has been agreed that nothing will be done about it.
- **Mitigated** – Action has been taken so the risk has been mitigated, either reducing the likelihood or reducing the impact.

Release Management (Release Demo)

One of the Agile Manifesto principles states that the primary measure of progress is working software. The IAE Agile Framework prescribes a Release Demo at the end of each release as well as at the end of each sprint, when Features are completed and can be presented via working software.

The Release Demo provides an integrated, program-level view of all new Features delivered by all the teams in the most recent iteration. The Release Demo lets the teams showcase their accomplishments and allows business stakeholders and customers to provide immediate feedback. The Release Demo also ensures that full and continuous integration takes place frequently. The System Team stages the Release Demo with support from individual development teams. At the end of a release, a retrospective is held immediately after the Release Demo to discuss how future releases can be executed more effectively.

The IAE Agile Framework supports three levels of Production Deployments, enabling continuous delivery to maximize business value delivery and minimize risk. The Release Management team will coordinate releases across the enterprise, working with the CCB for Feature level approvals. The Production Deployment plans enable the enterprise to prepare all the necessary technical and process activities such as putting Risk Mitigation plans in place to ensure a successful release.

Problem Solving and Continuous Improvement (Release Retrospective)

A principle in the Agile Manifesto states that at regular intervals, the team reflects on how to become more effective. IAE adheres to this principle by holding a Release Retrospective, also known as the Inspect and Adapt workshop, at the end of each release. A Release Demo is typically held prior to the Release Retrospective. During this ceremony, participants review the agreed upon quantitative metrics to evaluate the performance of the current release. This is followed by a retrospective on the release and a problem-solving workshop. Corrective action plans are devised and reviewed by the group. As a result of this workshop, teams come up with a set of improvement user stories that are added to the program backlog and incorporated into the next release planning session, thus assuring that action will be taken.

How to run a retrospective:

- A communication forum held at the conclusion of every Sprint/Release
- Agile teams come together to celebrate team successes
- Teams reflect on what to be improved, develop a SMART action plan to apply lessons learned going forward
- Identify top 3 impediments
- Create SMART action plan
- Scrum Master prioritizes actions and lessons learned based on Team direction
- Commit to track progress of the action plan

Discussion Points:

- What went well?
- What could be improved (lessons learned)?
- How to maintain what went well
- How to improve lessons learned

Retrospective concludes with agreed upon SMART Action Plan

- Specific – target a specific area for improvement.
- Measurable – quantify or at least suggest an indicator of progress.
- Assignable – specify who will do it.
- Realistic – state what results can realistically be achieved, given available resources.
- Time-related – specify when the result(s) can be achieved

Finding the Root Cause: The Five Whys

The Five Whys is a proven problem-solving technique used to explore the cause-and-effect relationships underlying a particular problem. By repeating the question, "Why?" five times, the nature of the problem, as well as the solution becomes clear. The key is to avoid assumptions and logic traps.

Example: The Problem: My car will not start.

- Why? – The battery is dead (first why)
- Why? – The alternator is not functioning (second why)
- Why? – The alternator belt has broken (third why)
- Why? – The alternator belt was well beyond its useful service life (fourth why)
- Why? – I have not been maintaining my car according to the recommended service schedule (fifth why, the root cause)

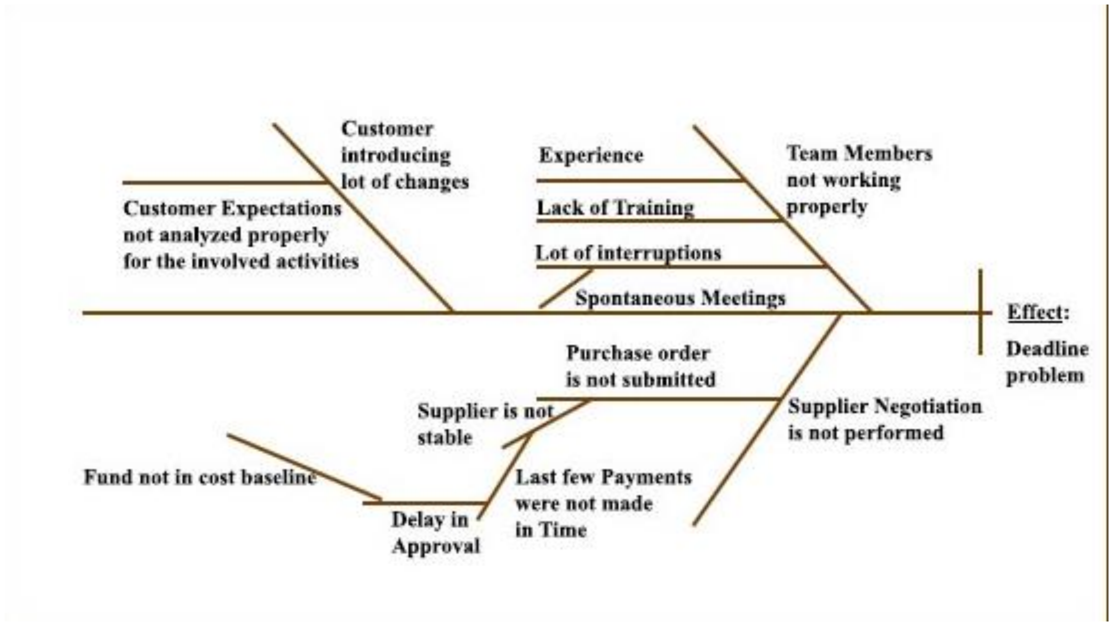

Customer introducing lot of changes
Customer Expectations not analyzed properly for the involved activities
Experience
Lack of Training
Lot of interruptions
Spontaneous Meetings
Team Members not working properly
Effect:
Deadline problem
Purchase order is not submitted
Supplier is not stable
Supplier Negotiation is not performed
Fund not in cost baseline
Delay in Approval
Last few Payments were not made in Time

FIGURE 16: ROOT CAUSE ANALYSIS – FISHBONE DIAGRAM

Appendix C3: Release Planning Checklist

Table 17: Release Planning Checklist (Day 1)

Release Planning Checklist Day 1	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7	Team 8	Team 9	Team 10
Check-In 1: Getting Started										
Do you understand the planning requirements?										
Do you know who your team is for the whole PI?										
Is your working space set up?										
Do you have a Program Manager & Scrum Master?										
Do you have access to the team members and stakeholders you need?										
Do you understand (and can you find) the vision that drives your backlog?										
Have you identified the velocity for each Sprint in your PI?										
Do you understand the architectural context, and whom to go to for questions?										
Do you understand which resources are shared (e.g., UX, Training, and Documentation) and whom to go to for questions?										
Do you understand the role of the System Team and DevOps, and whom to go to for questions?										
Check-In 2: Sprint Planning Progress										
Have hard dates been identified and represented on the Program Board?										
Are you identifying and estimating stories?										
Did you consider whether you need to allocate capacity to maintenance in your sprints?										

Is sprint story level planning in process?										
Have you identified any dependencies?										
Is there anything you need to discuss with other Scrum Masters? If so, stay for the "Meet After."										
Check-In 3: Team Objectives Progress										
Have you begun writing your Release Objectives?										
Have you finished allocating capacity (if necessary) to maintenance in your sprints?										
Have you identified most of your stories?										
Have you begun resolving dependencies with other teams?										
Are your dependencies represented on the Program Board?										
Are you identifying team and program risks?										
Do you anticipate completing your draft plan?										
Have all blocking issues that are impacting planning been removed?										
Are you discussing trade-offs and conflicting priorities with your business owners?										
Is there anything you need to discuss with other Scrum Masters? If so, stay for the "Meet After."										
Check-In 4: Draft Plan Readiness										
Are your Release Objectives and Stretch Objectives for the Release clear, legible, pithy and measurable (as per SMART Objectives)?										
Have you identified remaining team and program risks?										
Have you planned all sprints for your PI?										
Have most dependencies been identified and represented on the Program Board?										

Have all blocking issues that are impacting planning been removed?										
Are you ready to present your draft plan and risks?										
Is there anything you need to discuss with other Scrum Masters? If so, stay for the "Meet After."										

TABLE 18: RELEASE PLANNING CHECKLIST (DAY 2)

Release Planning Checklist Day 2	**Team 1**	**Team 2**	**Team 3**	**Team 4**	**Team 5**	**Team 6**	**Team 7**	**Team 8**	**Team 9**	**Team 10**
Check-In 1: Progress Check-In										
Do you understand how the management team feedback affects your team?										
Are you confident you have identified remaining dependencies based on the latest feedback, and have you represented them on the program board?										
Are you confident you have identified all hard dates based on the latest feedback and represented them on the program board?										
Are you confident you have identified your program risks based on the latest feedback?										
Have you re-planned all Sprints in the Release as necessary?										
Have you confirmed whether or not there is an opportunity to release before the end of the Release? (If there is, please indicate it on the Program Board in the Milestones/Events swim lane)										
Are your revised Release Objectives and Stretch Objectives clear, legible, pithy and more measurable (as per SMART Objectives)? Do they include dates where applicable?										
Have your business owners begun prioritizing your Release Objectives										

and Stretch Objectives?										
Do you anticipate completing your final plan?										
Have all blocking issues that are impacting planning been removed?										
Check-In 2: Final Plan Readiness										
Are all your NOs above YES now?										
Are your Release Objectives, Stretch Objectives, and business value finalized?										
Have you resolved all known remaining dependencies and represented them on the Program Board?										
Are your team risks mitigated?										
Are your program risks ready for review ("ROAM"ing)?										
Are you ready to present your final plan?										

APPENDIX C4: IAE PROGRAM MANAGER PROGRAM-LEVEL CHECKLIST

The following are the key tasks the IAE Program Manager needs to monitor/manage.

TABLE 19: IAE PROGRAM MANAGER PROGRAM CHECKLIST

IAE Program Manager Program-Level Checklist	Status
Understand the budget parameters for the upcoming fiscal period	
Understand how strategic themes influence the strategic direction of the ART	
Participate in discussions and development of the business case for Portfolio and Program Epics that affect your domain	
Create program vision	
Continuously develop and communicate the vision to the development teams	
Define and maintain the nonfunctional requirements to help assure the solution meets relevant standards and other system quality requirements	
Develop and communicate the roadmap for features for the next program increment	
Work with the System Architect to understand architectural work	
Create and maintain the Program Backlog	
Decompose program Epics to Features	
Manage the program Feature backlog and develop Feature acceptance criteria	
Define releases and program increments	
Participate in release management and solution validation	
Build an effective Product Management Team	

*IAE business and technical Program Managers wear multiple hats. They perform the roles of Epic Owner, Product Manager and Product Owner.

D. Team Information

Appendix D1: Team Roles and Responsibilities

The following are the key Roles and Responsibilities at the Team Level.

Table 20: Key Roles and Responsibilities (Team Level)

Role	Responsibility
Product Owner*	• Works with product management to plan releases • Defines and accepts User Stories • Defines User Story acceptance criteria • Maintains and prioritizes the Product Backlog • Clearly communicate the business case to the Team and Stakeholders
Scrum Master	• Runs team meetings, enforces Agile behavior • Removes impediments; protects the team from outside influence • Attends integration or Scrum of Scrums meetings • Creates the Task Board and Sprint Burndown Chart • Preserves the integrity and spirit of the IAE Agile framework
Scrum Team	• Develops and commits to Team's Release Objectives and Sprint Goals • Creates and refines User Stories and acceptance criteria • Implements, builds, tests, and deploys User Stories • Self-organizing and self-managing to accomplish Sprint goals
Stakeholders	• Participates in Sprint Planning, Review/Demo • Serves as one of the key sources of information while preparing product requirements (Epic/Feature/Story) • Remains engaged from definition to the completion of the Epic • Provides ongoing feedback and support • Collaborates with Product Owner • Reviews the Planned vs. Actual progress • Participates in Release Planning

* IAE business and technical Program Managers wear multiple hats. They perform the roles of Epic Owner, Product Manager and Product Owner.

APPENDIX D2: TEAM LEVEL PROCESS DETAILS

The following diagram represents the Team Level Process, which is at the lowest level in the framework.

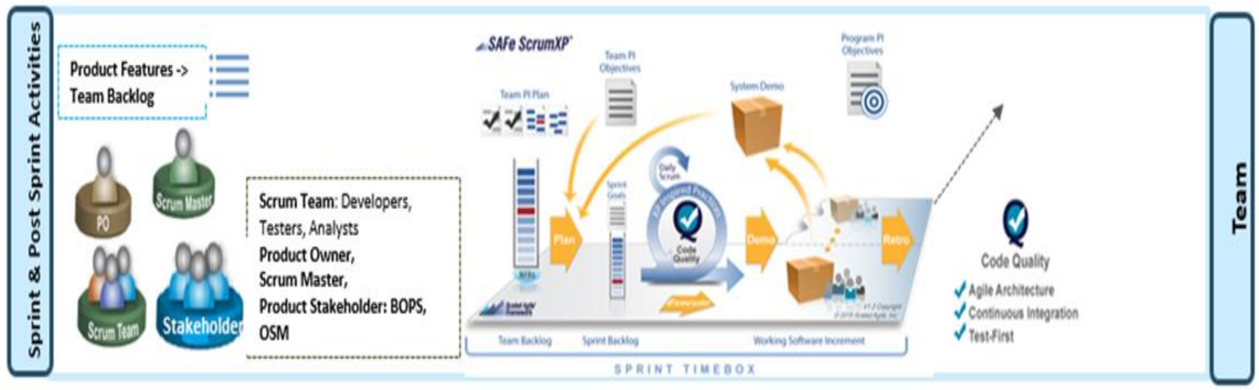


FIGURE 17: TEAM LEVEL PROCESS OVERVIEW

The team level in the IAE Agile Framework is composed of teams who are empowered to make local decisions and are accountable to deliver on the Program Commitments. The Product Owners collaborate with the Epic Owners and Architects to elaborate on the User Stories and create Acceptance Criteria. All the teams develop in cadence and synchronize to the Release Schedule. They produce valuable, fully-tested software increments every two weeks using Scrum project management and Agile Engineering Practices under the guidance of program vision, architecture and user experience resources. Teams work in a regular, synchronized cadence with one another to provide regular injection points to the system. This regular cadence is encompassed by the two-week sprint, which uses the current team backlog as input and includes the following:

- Planning for the upcoming sprint
- Execution of the sprint by developing and testing the agreed-to user stories,
- Demonstration of the resulting software, and
- The sprint retrospective.

TEAMS PRACTICES

- One way they do this is by developing User Stories incrementally by conducting multiple Define/Build/Test cycles during the course of the Sprint and not falling back on waterfall-style development methods.
- Teams must strive to create User Stories that are vertical slices of the technology stack and can be developed within a Sprint to create demonstrable software that increases business value delivery.
- Developing smaller pieces of functionality incrementally provides a quicker feedback mechanism because small chunks of working software can be continuously integrated into the larger system.
- This will also uncover dependencies across trains sooner.

- Agile Teams, in addition to the Dev Ops team, should also follow the Agile Engineering Practices in order to encourage the production of high quality code.
- These practices include Agile Architecture, Continuous Integration, Test-First, Refactoring, Pair Work, Collective Ownership and DevOps.
- These practices are described more thoroughly in the Agile Engineering Practices section.

TEAM PROCESS

The primary purpose of the team process is to ensure that quality software is developed and the product aligns with the program vision.

The following sections provide an overview of ceremonies in the Agile process. These ceremonies are structured with specific purpose and expected outcome, and they are designed to support continuous delivery of value as well as scaling Agile to the enterprise level. Please reference the Ceremonies table in Appendix A6 for additional details on the purpose, participants, inputs, outputs and frequency of each ceremony.

The purpose of team-level ceremonies is to provide a structure where implementation is done in a time-boxed and iterative fashion, in which Agile Teams deliver incremental value in the form of working software. IAE adheres to basic team-level practices based on SAFe, Scrum and eXtreme Programming (XP) practices. Agile Teams conduct Sprint Planning -> implementation -> demonstration -> retrospective to develop high quality products in two-week sprints.

It is highly recommended that all teams conform to Scrum at the team level. IAE will be conducting Scrum of Scrum meetings to discuss their work, focusing especially on areas of overlap, dependencies and integration.

BACKLOGS

Backlog Grooming ensures that the priorities align with the latest program direction and that the backlog is elaborated sufficiently to support successful Sprint Planning. The Agile Team is expected to support the Product Owner in grooming and refining the Team Backlog. Technical expertise will help establish accurate size estimates for items in the backlog. Large user stories may be decomposed into smaller stories and tasks.

TEAM BACKLOG

Team Backlog represents the collection of *all* the things a team needs to do to advance their portion of the system solution. It can contain User Stories, future Features, Technical Stories, tasks, defects, infrastructure work, spikes, refactors, and anything else a team needs to do. Stories can be broken down even further into tasks within the Story in order to make the development more manageable. Each Story must also contain a set of acceptance criteria that can be used to ensure that the Story delivers the intended benefits.

Sources in the “Team Backlog” can come from the program backlog, the team’s local context, or other stakeholders’ needs, as described below.

Program Backlog. During release planning, the Features that are planned for the release are broken into stories and tentatively allocated to individual upcoming Sprints in the team backlog. Features planned for upcoming releases can be stored in the team backlog. Also, the backlog can contain Stories for one team's portion of a program-level Feature that requires more than one team to complete.

Team Context. The team can also have a set of local Stories that are driven by the customer but do not come from the program backlog. This includes work such as maintenance, research, refactors and technology upgrades.

Other Stakeholders. The team backlog can also contain **S**tories that support other teams' and stakeholders' objectives.

The team must strike the proper balance between implementing new Stories while ensuring they take care of any internally-facing work that they need to complete.

The Product Owner is also the owner of the team backlog and is responsible for defining, prioritizing and maintaining it. IAE maintains the team backlog using the ALM tool.

Sprint Backlog

During Sprint Planning, teams select the highest priority items in the team backlog to implement in each Sprint. Stories coming from the program backlog should already be assigned a priority. Local stories can be prioritized using a value/size criterion, or even by applying full WSJF, if desired. Story point estimates must be provided for each Story.

At the Team level, individual teams decompose Features into Stories and Tasks in the Team Backlog, which can contain any item that the team needs to work on such as defects, infrastructure work or spikes. Spikes are a special type of research story used to gain the knowledge necessary to reduce the risk of a technical approach, better understand a requirement or increase the reliability of a Story estimate. The Agile Team meets at least once every Sprint to conduct Team Backlog Grooming. The Product Owner facilitates the Backlog Grooming session. During Team Backlog Grooming, the team does the following:

- Reviews the team backlog and implementation status
- Ensures that acceptance criteria exist for all stories
- Verifies story estimates
- Discusses the plan to build/maintain the development infrastructure and architectural runway, and
- Prepares for Sprint Planning.

Sprint Planning

The Product Owner attends Sprint Planning and presents the prioritized Team Backlog to the team. Each team picks high-priority User Stories from the Team Backlog and commits them to the Sprint backlog for execution. The team's backlog has already been partially pre-planned during Release Planning. Sprint Planning is typically time-boxed to four hours or fewer. The output of this process is the team's commitment to implement stories in the sprint backlog. The team also commits to achieve Sprint goals that support the current release objectives.

Sprint Planning is a key Scrum ceremony. The purposes of the meeting are listed below:

- The Product Owner attends Sprint Planning

- Present the prioritized Team Backlog
- Discuss the Stories, estimate Story Points, and commit to a set of User Stories for the Sprint.
- Attendees include the Product Owner, Scrum Master (who acts as the facilitator), Agile Team, and other stakeholders.
- The Sprint Planning session is essentially a matter of refining the Sprint plans that resulted from the release planning and previous Backlog Grooming session.
- The team begins by quantifying their capacity for the upcoming Sprint and developing the Sprint goals based on team and release objectives from the release planning session.
- Next, the team discusses each Story, develops acceptance criteria, determines size estimates and finalizes priorities for the Stories in the team backlog.
- User Stories are broken down into tasks, estimates are devised, dependencies are identified, and team members are assigned to each task until workloads are evenly distributed.
- The Team picks the high-priority backlog items and commits to the Sprint backlog for execution
- The Scrum Master ensures the team makes the commitment
- The Product Owner approves the Sprint plan.

TASKS

Stories can be decomposed into tasks to provide a breakdown of the individual requirements needed to complete the story. They facilitate coordination, estimation, status tracking and assigning individual responsibilities. Tasks provide value by exposing dependencies within the team as well as bottlenecks, resource availability, etc. Generally, a task is a small effort that can be completed within one day. If a task is too big to complete within two days, then the Story is too big and should be split. It is advisable t*o use the SMART criteria, as described below, when developing tasks.

- **S**pecific: A task needs to be specific enough that everyone can understand what's involved in it.
- **M**easureable: The key measure is, "Can we mark it as done?"
- **A**chievable: The task owner should expect to be able to achieve a task.
- **R**elevant: Every task should contribute to the story at hand.
- **T**ime-boxed: A task should be limited to a specific duration.

INFORMATION SHARING (DAILY SCRUM)

The Daily Scrum is designed to help the team set the context for the coming day's work. This meeting is time-boxed to 15 minutes to keep the discussions concise and relevant. The team can quickly share information on what was accomplished yesterday, what is planned for today, and report any impediments. The Scrum Master is responsible for helping the team resolve impediments.

THE SCRUM MEETING

- The daily Scrum is a 15-minute time-boxed meeting
- The development team synchronizes activities, communicates, and raises issues that hinder progress
- It is predetermined and held at the same time daily
- The team provides answers to the following 3 questions:
  - What have you accomplished since the last Scrum?

- What do you plan to accomplish before the next Scrum?
- What impediments do you have?

Benefits of Daily Scrum

- Less time spent in daily meetings
- Impediments immediately identified
- Daily snapshot on progress
- Early and frequent feedback
- Keeping the focus on identified goals of the Sprint
- Manages teams' ability to provide vital information to stakeholders→ resolve issues potentially before they may have to become a "**red flag**."

Demonstrate Sprint Deliverable (Sprint Demo)

The purpose of the Sprint Demo is to show the progress the team has made to the Product Owner and other stakeholders. The Sprint Demo, sometimes called the Team Demo or Sprint Review, takes place at the end of each two-week Sprint. The Sprint Demo starts with a quick review of the Sprint goals and metrics followed by a demonstration of each completed Story. After the demonstration, the team discusses which Stories were not completed and why they were not done. A retrospective is held immediately after the Sprint Demo to discuss how the team can be more effective going forward.

The Sprint Demo guidelines:

- The Product Owner attends the Sprint Demo
- The Product Owner accepts or rejects each User Story based on the DoD criterion.
- During this demo, teams demonstrate completed User Stories to the Product Owner and other stakeholders and review Sprint metrics.
- During the demonstration, the team should reference the Definition of Done for each user story. The demonstration is important because it is an opportunity to obtain valuable feedback from key stakeholders.
- Teams should start thinking about the demo during Sprint Planning in order to foster a more thorough understanding of the needed functionality.
- After the demonstration, teams should reflect on which Stories were not completed and why. The team should also address how well it is progressing toward release objectives.

Team Improvement Planning (Sprint Retrospective)

At the end of each Sprint, the team holds a retrospective to reflect on how to become more effective. This meeting is typically time-boxed to an hour for team members to discuss what went well, what didn't, and what the team can do better next time. The team reviews performance metrics and discusses any impediments and challenges faced during the past iteration. Root cause analysis is performed, and corrective actions are logged as user stories in the Team Backlog. The team picks one or two improvement items to target for the next Sprint. If necessary, the improvement items may become backlog items in the form of User Stories that are prioritized by the Product Owner with input from the team.

The Sprint Retrospective guidelines:

- During the Sprint Retrospective, teams discuss their practices and reflect on how to become more effective.
- This meeting is time-boxed to an hour or less.
- The whole team participates in the meeting, with the Scrum Master acting as facilitator.
- The meeting is composed of both quantitative and qualitative parts.
- For the quantitative review, the team determines whether they met the Sprint goals and then analyzes relevant metrics, including velocity.
- For the qualitative review, the team reviews the improvement backlog to determine if they met their previous improvement objectives.
- They also review the current Sprint to determine what went well, what didn’t go so well, and what they could do better next time.

Appendix D3: IAE Product Owner Team-Level Checklist

The following are the key tasks the Product Owner* needs to monitor and manage at the team level.

Table 21: IAE Product Owner Team Checklist

IAE Product Owner Team-Level Checklist	Status
Product Owner attends Sprint Planning	
Product Owner provides Sprint goal	
Product Owner approves Sprint Plan	
Product Owner provides story acceptance criteria	
Product Owner builds the product backlog	
Product Owner grooms the product backlog	
Product Owner maintains the product backlog	
Product Owner reviews and prioritizes the backlog for iteration planning	
Product Owner coordinates content dependencies with other Product Owners	
Product Owner approves final iteration plan	
Validate acceptance criteria and that each User Story has the appropriate, persistent acceptance tests, and otherwise meets its definition of done	
Attend sprint demos	
Product Owner accepts/rejects User Story based on DoD	
All team members attend retrospectives	
Product Owner inquires whether any root cause analysis (RCA) was performed	
Product Owner verifies RCA action plan	
Product Owner updates user story in team backlog with new RCA action plan	
Developers enter/maintain their tasks	

*IAE business and technical Program Managers wear multiple hats. They perform the roles of Epic Owner, Product Manager and Product Owner.